

EPICS-TwinCAT IOC

2.0

Generated by Doxygen 1.8.15

1 Module Index	1
1.1 Modules	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	11
5.1 File List	11
6 Module Documentation	13
6.1 Device support for TwinCAT/ADS	13
6.1.1 Detailed Description	16
6.1.2 Enumeration Type Documentation	16
6.1.2.1 epics_record_enum	16
6.1.3 Function Documentation	17
6.1.3.1 EpicsInterface()	17
6.1.3.2 get_callbackRequestPending()	18
6.1.3.3 linkRecord()	18
6.1.3.4 push()	19
6.1.4 Variable Documentation	19
6.1.4.1 ioscanpvt	20
6.1.4.2 isCallback	20
6.1.4.3 isPassive	20
6.1.4.4 the_register_devsup	20
6.1.4.5 value_alt_type	21
6.1.4.6 value_count	21
6.2 Functions called by the EPICS base	22
6.2.1 Detailed Description	22
6.2.2 Function Documentation	22
6.2.2.1 infoAlias()	22
6.2.2.2 infoList()	23
6.2.2.3 infoLoadRecords()	23
6.2.2.4 infoPrefix()	24
6.2.2.5 infoPrintVals()	24
6.2.2.6 infoSetScanRate()	25
6.2.2.7 tcAlias()	25
6.2.2.8 tcList()	25
6.2.2.9 tcLoadRecords()	26

6.2.2.10 tcMacro()	26
6.2.2.11 tcPrintVals()	27
6.2.2.12 tcSetScanRate()	27
6.3 Constants related to read/write scanning	28
6.3.1 Detailed Description	28
6.4 Classes for describing project information	29
6.4.1 Detailed Description	29
6.5 Classes for describing a structure element	30
6.5.1 Detailed Description	30
6.5.2 Typedef Documentation	30
6.5.2.1 dimension	30
6.5.2.2 dimensions	30
6.5.2.3 enum_map	31
6.5.2.4 enum_pair	31
6.5.2.5 item_list	31
6.6 Classes for describing a type	32
6.6.1 Detailed Description	32
6.6.2 Typedef Documentation	32
6.6.2.1 type_multimap	32
6.6.3 Enumeration Type Documentation	32
6.6.3.1 type_enum	32
6.7 Classes for describing a symbol	34
6.7.1 Detailed Description	34
6.7.2 Typedef Documentation	34
6.7.2.1 symbol_list	34
6.8 Classes for describing the parser	35
6.8.1 Detailed Description	35
6.9 XML tpy file constants	36
6.9.1 Detailed Description	37
6.10 OPC tpy file constants	38
6.10.1 Detailed Description	38
6.11 Option processing	39
6.11.1 Detailed Description	39
6.12 OPC related functions and classes	40
6.12.1 Detailed Description	40
6.12.2 Typedef Documentation	40
6.12.2.1 property_el	40
6.12.2.2 property_map	40
6.12.3 Enumeration Type Documentation	40
6.12.3.1 opc_enum	40
6.13 Classes for describing a parser argument	42
6.13.1 Detailed Description	42

6.13.2 Enumeration Type Documentation	42
6.13.2.1 process_tag_enum	42
6.13.2.2 process_type_enum	43
6.14 OPC property constants	44
6.14.1 Detailed Description	45
6.14.2 Variable Documentation	45
6.14.2.1 OPC_PROP_ALIAS	45
6.14.2.2 OPC_PROP_ALMAREAS	45
6.14.2.3 OPC_PROP_ALMCONDITION	45
6.14.2.4 OPC_PROP_ALMCOSV	46
6.14.2.5 OPC_PROP_ALMDB	46
6.14.2.6 OPC_PROP_ALMDEV	46
6.14.2.7 OPC_PROP_ALMFFSV	46
6.14.2.8 OPC_PROP_ALMH	47
6.14.2.9 OPC_PROP_ALMHELP	47
6.14.2.10 OPC_PROP_ALMH	47
6.14.2.11 OPC_PROP_ALMHHSV	47
6.14.2.12 OPC_PROP_ALMHSV	48
6.14.2.13 OPC_PROP_ALML	48
6.14.2.14 OPC_PROP_ALMLIMIT	48
6.14.2.15 OPC_PROP_ALMLL	48
6.14.2.16 OPC_PROP_ALMLLSV	49
6.14.2.17 OPC_PROP_ALMLSV	49
6.14.2.18 OPC_PROP_ALMOSV	49
6.14.2.19 OPC_PROP_ALMPRIMARYAREA	49
6.14.2.20 OPC_PROP_ALMROC	49
6.14.2.21 OPC_PROP_ALMSTAT	50
6.14.2.22 OPC_PROP_ALMUNSV	50
6.14.2.23 OPC_PROP_ALMZRSV	50
6.14.2.24 OPC_PROP_ALMZSV	50
6.14.2.25 OPC_PROP_AVI	50
6.14.2.26 OPC_PROP_BGC	51
6.14.2.27 OPC_PROP_BLINK	51
6.14.2.28 OPC_PROP_BMP	51
6.14.2.29 OPC_PROP_CDT	51
6.14.2.30 OPC_PROP_CLOSE	51
6.14.2.31 OPC_PROP_DESC	52
6.14.2.32 OPC_PROP_DTYP	52
6.14.2.33 OPC_PROP_FFST	52
6.14.2.34 OPC_PROP_FGC	52
6.14.2.35 OPC_PROP_HIEU	53
6.14.2.36 OPC_PROP_HIRANGE	53

6.14.2.37 OPC_PROP_HTML	53
6.14.2.38 OPC_PROP_INOUT	53
6.14.2.39 OPC_PROP_INPUT	54
6.14.2.40 OPC_PROP_LOEU	54
6.14.2.41 OPC_PROP_LORANGE	54
6.14.2.42 OPC_PROP_OPEN	54
6.14.2.43 OPC_PROP_OUTPUT	55
6.14.2.44 OPC_PROP_PINI	55
6.14.2.45 OPC_PROP_PLNAME	55
6.14.2.46 OPC_PROP_PREC	55
6.14.2.47 OPC_PROP_QUALITY	56
6.14.2.48 OPC_PROP_RECTYPE	56
6.14.2.49 OPC_PROP_RIGHTS	56
6.14.2.50 OPC_PROP_SCANRATE	56
6.14.2.51 OPC_PROP_SERVER	56
6.14.2.52 OPC_PROP SND	57
6.14.2.53 OPC_PROP_TIME	57
6.14.2.54 OPC_PROP_TIMEZONE	57
6.14.2.55 OPC_PROP_TSE	57
6.14.2.56 OPC_PROP_UNIT	57
6.14.2.57 OPC_PROP_VALUE	58
6.14.2.58 OPC_PROP_ZRST	58
6.15 utility functions and classes	59
6.15.1 Detailed Description	59
6.15.2 Typedef Documentation	59
6.15.2.1 stringcase	59
6.15.2.2 wstringcase	60
6.15.3 Function Documentation	60
6.15.3.1 split_string()	60
6.15.3.2 strncasecmp()	61
6.15.3.3 trim_space() [1/2]	61
6.15.3.4 trim_space() [2/2]	62
6.15.3.5 wcsncasewcmp()	62
6.16 Utility functions and classes	64
6.16.1 Detailed Description	64
6.17 Classes for describing TC symbol	65
6.17.1 Detailed Description	65
6.18 Classes for managing groups of TC symbols	66
6.18.1 Detailed Description	66
6.19 Utility functions and classes	67
6.19.1 Detailed Description	67
6.19.2 Typedef Documentation	67

6.19.2.1 replacement_table	67
6.19.3 Enumeration Type Documentation	67
6.19.3.1 case_type	67
6.19.3.2 io_filestat	68
6.19.3.3 tc_epics_conv	68
6.20 Classes for converting a parsed tpy	69
6.20.1 Detailed Description	69
6.20.2 Typedef Documentation	69
6.20.2.1 filename_set	69
6.20.2.2 macro_list	70
6.20.2.3 macro_stack	70
6.20.3 Enumeration Type Documentation	70
6.20.3.1 device_support_type	70
6.20.3.2 listing_type	70
6.20.3.3 macrofile_type	71
6.21 EPICS maximum length constants	72
6.21.1 Detailed Description	72
6.21.2 Variable Documentation	72
6.21.2.1 MAX_EPICS_CHANNEL	72
6.21.2.2 MAX_EPICS_DESC	72
6.21.2.3 MAX_EPICS_UNIT	72
6.22 EPICS record field names	73
6.22.1 Detailed Description	73
6.22.2 Variable Documentation	73
6.22.2.1 EPICS_DB_COSV	73
6.22.2.2 EPICS_DB_DESC	74
6.22.2.3 EPICS_DB_DRVH	74
6.22.2.4 EPICS_DB_DRVL	74
6.22.2.5 EPICS_DB_DTYP	74
6.22.2.6 EPICS_DB_EGU	75
6.22.2.7 EPICS_DB_HHSV	75
6.22.2.8 EPICS_DB_HIGH	75
6.22.2.9 EPICS_DB_HIHI	75
6.22.2.10 EPICS_DB_HOPR	76
6.22.2.11 EPICS_DB_HSV	76
6.22.2.12 EPICS_DB_HYST	76
6.22.2.13 EPICS_DB_INP	76
6.22.2.14 EPICS_DB_LLSV	77
6.22.2.15 EPICS_DB_LOLO	77
6.22.2.16 EPICS_DB_LOPR	77
6.22.2.17 EPICS_DB_LOW	77
6.22.2.18 EPICS_DB_LSV	78

6.22.2.19 EPICS_DB_MAJOR	78
6.22.2.20 EPICS_DB_MINOR	78
6.22.2.21 EPICS_DB_NOALARM	78
6.22.2.22 EPICS_DB_ONAM	79
6.22.2.23 EPICS_DB_OSV	79
6.22.2.24 EPICS_DB_OUT	79
6.22.2.25 EPICS_DB_PINI	79
6.22.2.26 EPICS_DB_PREC	80
6.22.2.27 EPICS_DB_SCAN	80
6.22.2.28 EPICS_DB_TSE	80
6.22.2.29 EPICS_DB_UNSV	80
6.22.2.30 EPICS_DB_ZNAM	81
6.22.2.31 EPICS_DB_ZRST	81
6.22.2.32 EPICS_DB_ZRSV	81
6.22.2.33 EPICS_DB_ZRVL	81
6.22.2.34 EPICS_DB_ZSV	81
6.23 Lists of EPICS record field names	82
6.23.1 Detailed Description	82
6.23.2 Variable Documentation	82
6.23.2.1 EPICS_DB_ALLOWED	82
6.23.2.2 EPICS_DB_FORBIDDEN	82
6.23.2.3 EPICS_DB_NUMVAL	82
6.24 LIGO related constants	83
6.24.1 Detailed Description	83
6.24.2 Variable Documentation	83
6.24.2.1 LIGODAQ_DATATYPE_DEFAULT	83
6.24.2.2 LIGODAQ_DATATYPE_FLOAT	83
6.24.2.3 LIGODAQ_DATATYPE_INT32	83
6.24.2.4 LIGODAQ_DATATYPE_NAME	84
6.24.2.5 LIGODAQ_INI_HEADER	84
6.24.2.6 LIGODAQ_UNIT_DEFAULT	84
6.24.2.7 LIGODAQ_UNIT_NAME	84
6.24.2.8 LIGODAQ_UNIT_NONE	84
7 Namespace Documentation	85
7.1 DevInfo Namespace Reference	85
7.1.1 Detailed Description	86
7.1.2 Typedef Documentation	86
7.1.2.1 filename_rule_pair	86
7.1.3 Function Documentation	86
7.1.3.1 linkInfoRecord()	86
7.2 DevTc Namespace Reference	87

7.2.1 Detailed Description	89
7.2.2 Function Documentation	89
7.2.2.1 linkRecord()	89
7.2.2.2 linkTcRecord()	90
7.2.2.3 outRecordCallback()	91
7.3 EpicsTpy Namespace Reference	91
7.3.1 Detailed Description	93
7.4 InfoPlc Namespace Reference	94
7.4.1 Detailed Description	95
7.4.2 Typedef Documentation	95
7.4.2.1 stat_list	95
7.5 ParseTpy Namespace Reference	95
7.5.1 Detailed Description	98
7.5.2 Function Documentation	98
7.5.2.1 compareNamesWoNamespace()	98
7.5.2.2 get_decoration()	98
7.5.2.3 get_pointer()	98
7.6 ParseUtil Namespace Reference	99
7.6.1 Detailed Description	100
7.7 plc Namespace Reference	101
7.7.1 Detailed Description	102
7.7.2 Typedef Documentation	102
7.7.2.1 BasePLCList	102
7.7.2.2 BasePLCPtr	103
7.7.2.3 BaseRecordList	103
7.7.2.4 BaseRecordPtr	103
7.7.2.5 InterfacePtr	103
7.7.3 Enumeration Type Documentation	103
7.7.3.1 access_rights_enum	103
7.7.3.2 data_type_enum	104
7.7.4 Function Documentation	104
7.7.4.1 reset_and_read() [1/3]	104
7.7.4.2 reset_and_read() [2/3]	105
7.7.4.3 reset_and_read() [3/3]	105
7.7.4.4 ScannerProc()	106
7.7.4.5 scannerThread()	107
7.7.4.6 write_and_test() [1/3]	108
7.7.4.7 write_and_test() [2/3]	108
7.7.4.8 write_and_test() [3/3]	108
7.8 TcComms Namespace Reference	109
7.8.1 Detailed Description	110
7.8.2 Function Documentation	110

7.8.2.1 ADScallback()	110
7.8.2.2 compByOffset()	111
7.8.2.3 errorPrintf()	112
7.8.2.4 RouterCall()	112
8 Class Documentation	113
8.1 ParseTpy::ads_routing_info Class Reference	113
8.1.1 Detailed Description	114
8.1.2 Member Function Documentation	114
8.1.2.1 get() [1/2]	114
8.1.2.2 get() [2/2]	115
8.1.2.3 set()	115
8.2 TcComms::AmsRouterNotification Class Reference	116
8.2.1 Detailed Description	116
8.2.2 Friends And Related Function Documentation	116
8.2.2.1 RouterCall	117
8.3 std::atomic< string > Class Template Reference	117
8.3.1 Detailed Description	118
8.4 std::atomic< wstring > Class Template Reference	118
8.4.1 Detailed Description	119
8.5 std::atomic_string< stringT > Class Template Reference	120
8.5.1 Detailed Description	121
8.5.2 Member Function Documentation	121
8.5.2.1 operator=()	121
8.6 ParseTpy::base_record Class Reference	122
8.6.1 Detailed Description	123
8.6.2 Constructor & Destructor Documentation	123
8.6.2.1 base_record() [1/4]	123
8.6.2.2 base_record() [2/4]	124
8.6.2.3 base_record() [3/4]	124
8.6.2.4 base_record() [4/4]	124
8.7 InfoPlc::BaseInfoItem Class Reference	125
8.7.1 Detailed Description	126
8.7.2 Member Function Documentation	126
8.7.2.1 get_info()	127
8.7.2.2 setup()	127
8.8 plc::BasePLC Class Reference	128
8.8.1 Detailed Description	131
8.8.2 Member Function Documentation	131
8.8.2.1 add() [1/2]	131
8.8.2.2 add() [2/2]	131
8.8.2.3 erase()	132

8.8.2.4 find()	132
8.8.2.5 for_each()	133
8.8.2.6 get_next()	133
8.8.2.7 get_timestamp_unix()	134
8.8.2.8 plc_data_set_valid()	134
8.8.2.9 reserve()	135
8.8.2.10 start()	135
8.8.2.11 user_data_set_valid()	135
8.8.3 Member Data Documentation	136
8.8.3.1 records	136
8.9 plc::BaseRecord Class Reference	137
8.9.1 Detailed Description	139
8.9.2 Constructor & Destructor Documentation	140
8.9.2.1 BaseRecord() [1/2]	140
8.9.2.2 BaseRecord() [2/2]	140
8.9.3 Member Function Documentation	140
8.9.3.1 PlcGetValid()	140
8.9.3.2 PlcPush()	141
8.9.3.3 PlcRead() [1/3]	142
8.9.3.4 PlcRead() [2/3]	143
8.9.3.5 PlcRead() [3/3]	144
8.9.3.6 PlcReadBinary()	145
8.9.3.7 PlcSetValid()	146
8.9.3.8 PlcWrite() [1/3]	146
8.9.3.9 PlcWrite() [2/3]	147
8.9.3.10 PlcWrite() [3/3]	148
8.9.3.11 PlcWriteBinary()	148
8.9.3.12 UserGetValid()	149
8.9.3.13 UserPush()	149
8.9.3.14 UserRead() [1/3]	150
8.9.3.15 UserRead() [2/3]	151
8.9.3.16 UserRead() [3/3]	152
8.9.3.17 UserReadBinary()	152
8.9.3.18 UserSetValid()	153
8.9.3.19 UserWrite() [1/3]	154
8.9.3.20 UserWrite() [2/3]	155
8.9.3.21 UserWrite() [3/3]	155
8.9.3.22 UserWriteBinary()	156
8.10 ParseUtil::bit_location Class Reference	157
8.10.1 Detailed Description	158
8.11 std::case_char_traits Struct Reference	158
8.11.1 Detailed Description	159

8.11.2 Member Function Documentation	159
8.11.2.1 compare()	159
8.11.2.2 eq()	160
8.11.2.3 lt()	160
8.11.2.4 ne()	160
8.12 std::case_wchar_traits Struct Reference	161
8.12.1 Detailed Description	161
8.12.2 Member Function Documentation	162
8.12.2.1 compare()	162
8.12.2.2 eq()	162
8.12.2.3 lt()	163
8.12.2.4 ne()	163
8.13 ParseTpy::compiler_info Class Reference	163
8.13.1 Detailed Description	165
8.14 TcComms::DataPar Struct Reference	165
8.14.1 Detailed Description	165
8.15 plc::DataValue Class Reference	166
8.15.1 Detailed Description	168
8.15.2 Constructor & Destructor Documentation	169
8.15.2.1 DataValue()	169
8.15.3 Member Function Documentation	169
8.15.3.1 GetValid()	169
8.15.3.2 Init()	170
8.15.3.3 PlcGetValid()	171
8.15.3.4 PlcRead() [1/4]	172
8.15.3.5 PlcRead() [2/4]	172
8.15.3.6 PlcRead() [3/4]	173
8.15.3.7 PlcRead() [4/4]	173
8.15.3.8 PlcReadBinary()	174
8.15.3.9 PlcSetValid()	175
8.15.3.10 PlcWrite() [1/4]	176
8.15.3.11 PlcWrite() [2/4]	176
8.15.3.12 PlcWrite() [3/4]	177
8.15.3.13 PlcWrite() [4/4]	177
8.15.3.14 PlcWriteBinary()	178
8.15.3.15 Read() [1/5]	179
8.15.3.16 Read() [2/5]	180
8.15.3.17 Read() [3/5]	180
8.15.3.18 Read() [4/5]	181
8.15.3.19 Read() [5/5]	182
8.15.3.20 ReadBinary()	182
8.15.3.21 SetValid()	183

8.15.3.22 UserGetValid()	184
8.15.3.23 UserRead() [1/4]	185
8.15.3.24 UserRead() [2/4]	185
8.15.3.25 UserRead() [3/4]	186
8.15.3.26 UserRead() [4/4]	186
8.15.3.27 UserReadBinary()	187
8.15.3.28 UserSetValid()	188
8.15.3.29 UserWrite() [1/4]	189
8.15.3.30 UserWrite() [2/4]	189
8.15.3.31 UserWrite() [3/4]	190
8.15.3.32 UserWrite() [4/4]	190
8.15.3.33 UserWriteBinary()	191
8.15.3.34 Write() [1/5]	192
8.15.3.35 Write() [2/5]	193
8.15.3.36 Write() [3/5]	194
8.15.3.37 Write() [4/5]	194
8.15.3.38 Write() [5/5]	195
8.15.3.39 WriteBinary()	195
8.15.4 Member Data Documentation	197
8.15.4.1 mysize	197
8.16 plc::DataValueTraits< T > Struct Template Reference	198
8.16.1 Detailed Description	199
8.16.2 Member Data Documentation	199
8.16.2.1 data_enum	199
8.17 plc::DataValueTypeDef Struct Reference	199
8.17.1 Detailed Description	201
8.18 DevTc::devTcDefIn< RecType > Struct Template Reference	201
8.18.1 Detailed Description	203
8.19 DevTc::devTcDefIo< RecType > Struct Template Reference	203
8.19.1 Detailed Description	204
8.20 DevTc::devTcDefOut< RecType > Struct Template Reference	205
8.20.1 Detailed Description	206
8.21 DevTc::devTcDefWaveformIn< RecType > Struct Template Reference	206
8.21.1 Detailed Description	207
8.22 EpicsTpy::epics_conversion Class Reference	207
8.22.1 Detailed Description	209
8.22.2 Constructor & Destructor Documentation	209
8.22.2.1 epics_conversion() [1/3]	209
8.22.2.2 epics_conversion() [2/3]	209
8.22.2.3 epics_conversion() [3/3]	210
8.22.3 Member Function Documentation	210
8.22.3.1 getopt()	210

8.22.3.2 <code>to_epics()</code>	212
8.23 <code>EpicsTpy::epics_db_processing</code> Class Reference	213
8.23.1 Detailed Description	215
8.23.2 Constructor & Destructor Documentation	215
8.23.2.1 <code>epics_db_processing()</code>	215
8.23.3 Member Function Documentation	216
8.23.3.1 <code>getopt()</code>	216
8.23.3.2 <code>my getopt()</code>	217
8.23.3.3 <code>operator()()</code>	218
8.23.3.4 <code>process_field_alarm()</code>	220
8.23.3.5 <code>process_field_numeric()</code> [1/3]	221
8.23.3.6 <code>process_field_numeric()</code> [2/3]	222
8.23.3.7 <code>process_field_numeric()</code> [3/3]	223
8.23.3.8 <code>process_field_string()</code>	224
8.24 <code>DevInfo::epics_info_db_processing</code> Class Reference	225
8.24.1 Detailed Description	226
8.24.2 Member Function Documentation	226
8.24.2.1 <code>operator()()</code>	226
8.25 <code>EpicsTpy::epics_list_processing</code> Class Reference	227
8.25.1 Detailed Description	229
8.25.2 Constructor & Destructor Documentation	229
8.25.2.1 <code>epics_list_processing()</code> [1/2]	229
8.25.2.2 <code>epics_list_processing()</code> [2/2]	230
8.25.3 Member Function Documentation	230
8.25.3.1 <code>getopt()</code>	230
8.25.3.2 <code>my getopt()</code>	231
8.25.3.3 <code>operator()()</code>	232
8.26 <code>EpicsTpy::epics_macrofiles_processing</code> Class Reference	234
8.26.1 Detailed Description	237
8.26.2 Constructor & Destructor Documentation	237
8.26.2.1 <code>epics_macrofiles_processing()</code> [1/2]	237
8.26.2.2 <code>epics_macrofiles_processing()</code> [2/2]	237
8.26.3 Member Function Documentation	238
8.26.3.1 <code>getopt()</code>	238
8.26.3.2 <code>my getopt()</code>	239
8.26.3.3 <code>operator()()</code>	240
8.26.3.4 <code>to_filename()</code>	241
8.27 <code>DevTc::epics_record_traits< RecType ></code> Struct Template Reference	242
8.27.1 Detailed Description	243
8.28 <code>DevTc::epics_tc_db_processing</code> Class Reference	243
8.28.1 Detailed Description	245
8.28.2 Member Function Documentation	245

8.28.2.1 operator()()	245
8.28.2.2 process_list()	246
8.28.2.3 process_lists()	247
8.28.2.4 process_macro()	247
8.28.2.5 process_macros()	248
8.29 DevTc::EpicsInterface Class Reference	248
8.29.1 Detailed Description	250
8.30 std::std::hash< std::stringcase > Struct Template Reference	251
8.30.1 Detailed Description	251
8.30.2 Member Function Documentation	251
8.30.2.1 operator()()	251
8.31 std::std::hash< std::wstringcase > Struct Template Reference	252
8.31.1 Detailed Description	252
8.31.2 Member Function Documentation	252
8.31.2.1 operator()()	252
8.32 InfoPlc::HistogramInfoItem Class Reference	253
8.32.1 Detailed Description	254
8.33 InfoPlc::HistoryInfoItem Class Reference	254
8.33.1 Detailed Description	255
8.33.2 Member Function Documentation	256
8.33.2.1 setup()	256
8.34 InfoPlc::InfoInterface Class Reference	256
8.34.1 Detailed Description	258
8.35 InfoPlc::InfoPLC Class Reference	258
8.35.1 Detailed Description	259
8.35.2 Member Function Documentation	259
8.35.2.1 process_info() [1/2]	259
8.35.2.2 process_info() [2/2]	260
8.36 DevInfo::InfoRegisterToLocShell Class Reference	261
8.36.1 Detailed Description	261
8.37 plc::Interface Class Reference	262
8.37.1 Detailed Description	263
8.37.2 Constructor & Destructor Documentation	263
8.37.2.1 Interface()	263
8.38 ParseTpy::item_record Class Reference	263
8.38.1 Detailed Description	264
8.39 EpicsTpy::macro_info Struct Reference	265
8.39.1 Detailed Description	265
8.40 EpicsTpy::macro_record Struct Reference	265
8.40.1 Detailed Description	266
8.41 ParseUtil::memory_location Class Reference	266
8.41.1 Detailed Description	267

8.41.2 Constructor & Destructor Documentation	267
8.41.2.1 memory_location() [1/2]	267
8.41.2.2 memory_location() [2/2]	268
8.41.3 Member Function Documentation	268
8.41.3.1 get()	268
8.41.3.2 set()	269
8.41.3.3 set_section()	270
8.42 EpicsTpy::multi_io_support Class Reference	270
8.42.1 Detailed Description	272
8.42.2 Constructor & Destructor Documentation	272
8.42.2.1 multi_io_support()	272
8.42.3 Member Function Documentation	273
8.42.3.1 getopt()	273
8.43 ParseUtil::opc_list Class Reference	274
8.43.1 Detailed Description	275
8.44 ParseUtil::optarg Class Reference	275
8.44.1 Detailed Description	276
8.44.2 Constructor & Destructor Documentation	276
8.44.2.1 optarg()	276
8.44.3 Member Function Documentation	277
8.44.3.1 parse()	277
8.45 ParseTpy::parserinfo_type Class Reference	278
8.45.1 Detailed Description	280
8.45.2 Member Data Documentation	280
8.45.2.1 name_parse	280
8.46 ParseUtil::process_arg Class Reference	281
8.46.1 Detailed Description	282
8.46.2 Constructor & Destructor Documentation	282
8.46.2.1 process_arg()	282
8.46.3 Member Function Documentation	283
8.46.3.1 get()	283
8.46.3.2 get_full()	283
8.47 ParseTpy::project_record Class Reference	284
8.47.1 Detailed Description	285
8.48 DevTc::register_devsup Class Reference	285
8.48.1 Detailed Description	286
8.49 DevInfo::register_info_devsup Class Reference	287
8.49.1 Detailed Description	287
8.50 EpicsTpy::replacement_rules Class Reference	288
8.50.1 Detailed Description	289
8.51 plc::scanner_thread_args Struct Reference	289
8.51.1 Detailed Description	289

8.52 InfoPlc::SimpleInfoItem Class Reference	290
8.52.1 Detailed Description	291
8.52.2 Member Function Documentation	291
8.52.2.1 setup()	291
8.53 EpicsTpy::split_io_support Class Reference	292
8.53.1 Detailed Description	294
8.53.2 Constructor & Destructor Documentation	294
8.53.2.1 split_io_support() [1/2]	294
8.53.2.2 split_io_support() [2/2]	295
8.53.3 Member Function Documentation	295
8.53.3.1 getopt()	295
8.53.3.2 increment()	296
8.53.3.3 operator=()	297
8.54 InfoPlc::stat_value Struct Reference	298
8.54.1 Detailed Description	298
8.55 ParseTpy::symbol_record Class Reference	299
8.55.1 Detailed Description	300
8.56 syminfo_processing Class Reference	300
8.56.1 Detailed Description	300
8.57 plc::System Class Reference	300
8.57.1 Detailed Description	301
8.57.2 Member Function Documentation	302
8.57.2.1 add() [1/2]	302
8.57.2.2 add() [2/2]	302
8.57.2.3 for_each()	302
8.58 ParseUtil::tag_processing Class Reference	303
8.58.1 Detailed Description	304
8.58.2 Constructor & Destructor Documentation	304
8.58.2.1 tag_processing() [1/2]	304
8.58.2.2 tag_processing() [2/2]	305
8.58.3 Member Function Documentation	305
8.58.3.1 getopt()	306
8.59 TcComms::TCatInterface Class Reference	307
8.59.1 Detailed Description	309
8.59.2 Constructor & Destructor Documentation	309
8.59.2.1 TCatInterface()	309
8.59.3 Member Function Documentation	310
8.59.3.1 printTCatVal()	310
8.60 TcComms::TcPLC Class Reference	311
8.60.1 Detailed Description	314
8.60.2 Member Function Documentation	314
8.60.2.1 closePort()	314

8.60.2.2 optimizeRequests()	315
8.60.2.3 set_addr()	315
8.60.2.4 set_ads_state()	316
8.60.2.5 update_scanner()	316
8.60.3 Friends And Related Function Documentation	317
8.60.3.1 ADSCallback	317
8.60.4 Member Data Documentation	317
8.60.4.1 cyclesLeft	317
8.61 TcComms::tcProcWrite Class Reference	317
8.61.1 Detailed Description	318
8.61.2 Member Function Documentation	319
8.61.2.1 add()	319
8.61.2.2 read_ptr()	320
8.61.3 Member Data Documentation	320
8.61.3.1 req	320
8.62 DevTc::tcRegisterTolocShell Class Reference	321
8.62.1 Detailed Description	321
8.63 InfoPlc::TimeInfoItem Class Reference	321
8.63.1 Detailed Description	322
8.63.2 Member Function Documentation	323
8.63.2.1 start()	323
8.64 ParseTpy::tpy_file Class Reference	323
8.64.1 Detailed Description	325
8.64.2 Member Function Documentation	325
8.64.2.1 parse_finish()	325
8.64.2.2 process_array()	325
8.64.2.3 process_symbols()	327
8.64.2.4 process_type_tree() [1/3]	328
8.64.2.5 process_type_tree() [2/3]	329
8.64.2.6 process_type_tree() [3/3]	331
8.65 DevTc::epics_record_traits< RecType >::traits_type Struct Reference	333
8.65.1 Detailed Description	333
8.66 ParseTpy::type_map Class Reference	333
8.66.1 Detailed Description	334
8.67 ParseTpy::type_record Class Reference	334
8.67.1 Detailed Description	336
8.68 ParseUtil::variable_name Class Reference	336
8.68.1 Detailed Description	337
9 File Documentation	339
9.1 atomic_string.h File Reference	339
9.1.1 Detailed Description	340

9.2 devInfo.cpp File Reference	340
9.2.1 Detailed Description	341
9.3 devInfo.h File Reference	342
9.3.1 Detailed Description	343
9.4 devTc.cpp File Reference	343
9.4.1 Detailed Description	344
9.5 devTc.h File Reference	344
9.5.1 Detailed Description	346
9.6 drvInfo.cpp File Reference	347
9.6.1 Detailed Description	348
9.7 drvInfo.h File Reference	348
9.7.1 Detailed Description	349
9.8 drvTc.cpp File Reference	349
9.8.1 Detailed Description	350
9.9 drvTc.h File Reference	350
9.9.1 Detailed Description	351
9.10 EpicsDbGen.cpp File Reference	351
9.10.1 Detailed Description	352
9.10.2 Function Documentation	352
9.10.2.1 main()	352
9.11 infoPlc.cpp File Reference	353
9.11.1 Detailed Description	354
9.12 infoPlc.h File Reference	354
9.12.1 Detailed Description	355
9.13 infoPlcTemplate.h File Reference	356
9.13.1 Detailed Description	356
9.14 iocMain.cpp File Reference	356
9.14.1 Detailed Description	357
9.14.2 Function Documentation	357
9.14.2.1 main()	357
9.15 ParseTpy.cpp File Reference	358
9.15.1 Detailed Description	358
9.16 ParseTpy.h File Reference	359
9.16.1 Detailed Description	360
9.17 ParseTpyConst.h File Reference	360
9.17.1 Detailed Description	362
9.18 ParseTpyInfo.cpp File Reference	363
9.18.1 Detailed Description	363
9.18.2 Function Documentation	363
9.18.2.1 main()	363
9.19 ParseTpyTemplate.h File Reference	364
9.19.1 Detailed Description	364

9.20 ParseUtil.cpp File Reference	365
9.20.1 Detailed Description	365
9.21 ParseUtil.h File Reference	365
9.21.1 Detailed Description	366
9.22 ParseUtilConst.h File Reference	367
9.22.1 Detailed Description	368
9.23 plcBase.cpp File Reference	369
9.23.1 Detailed Description	369
9.24 plcBase.h File Reference	370
9.24.1 Detailed Description	371
9.25 plcBaseTemplate.h File Reference	372
9.25.1 Detailed Description	373
9.26 stdafx.cpp File Reference	373
9.26.1 Detailed Description	373
9.27 stdafx.h File Reference	373
9.27.1 Detailed Description	374
9.28 stringcase.h File Reference	374
9.28.1 Detailed Description	375
9.29 stringcase_hash.h File Reference	375
9.29.1 Detailed Description	376
9.30 tcComms.cpp File Reference	376
9.30.1 Detailed Description	377
9.31 tcComms.h File Reference	377
9.31.1 Detailed Description	378
9.32 TpyToEpics.cpp File Reference	379
9.32.1 Detailed Description	379
9.33 TpyToEpics.h File Reference	379
9.33.1 Detailed Description	381
9.34 TpyToEpicsConst.h File Reference	381
9.34.1 Detailed Description	383
Index	385

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Device support for TwinCAT/ADS	13
Functions called by the EPICS base	22
Constants related to read/write scanning	28
Classes for describing project information	29
Classes for describing a structure element	30
Classes for describing a type	32
Classes for describing a symbol	34
Classes for describing the parser	35
XML tpy file constants	36
OPC tpy file constants	38
Option processing	39
OPC related functions and classes	40
Classes for describing a parser argument	42
OPC property constants	44
utility functions and classes	59
Utility functions and classes	64
Classes for describing TC symbol	65
Classes for managing groups of TC symbols	66
Utility functions and classes	67
Classes for converting a parsed tpy	69
EPICS maximum length constants	72
EPICS record field names	73
Lists of EPICS record field names	82
LIGO related constants	83

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

DevInfo	Namespace for info device support	85
DevTc	Namespace for info device support	87
EpicsTpy	Namespace for tpy-db conversion	91
InfoPlc	Namespace for Info communication	94
ParseTpy	Namespace for parsing	95
ParseUtil	Namespace for parsing utilities	99
plc	Namespace for abstract plc functionality	101
TcComms	Namespace for TCat communication	109

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ParseTpy::ads_routing_info	113
ParseTpy::project_record	284
TcComms::AmsRouterNotification	116
std::atomic_string< stringT >	120
std::atomic_string< string >	120
std::atomic< string >	117
std::atomic_string< wstring >	120
std::atomic< wstring >	118
ParseTpy::base_record	122
ParseTpy::item_record	263
ParseTpy::symbol_record	299
ParseTpy::type_record	334
InfoPlc::BaseInfoItem	125
InfoPlc::HistogramInfoItem	253
InfoPlc::HistoryInfoItem	254
InfoPlc::SimpleInfoItem	290
InfoPlc::TimeInfoItem	321
plc::BasePLC	128
InfoPlc::InfoPLC	258
TcComms::TcPLC	311
ParseUtil::bit_location	157
ParseTpy::item_record	263
ParseTpy::type_record	334
char_traits	
std::case_char_traits	158
std::case_wchar_traits	161
ParseTpy::compiler_info	163
ParseTpy::project_record	284
TcComms::DataPar	165
plc::DataValueTraits< T >	198
plc::DataValueTypeDef	199
plc::BaseRecord	137
plc::DataValue	166

DevTc::devTcDeflo< RecType >	203
DevTc::devTcDefIn< RecType >	201
DevTc::devTcDefOut< RecType >	205
DevTc::devTcDefWaveformIn< RecType >	206
DevTc::epics_record_traits< RecType >	242
std::std::hash< std::stringcase >	251
std::std::hash< std::wstringcase >	252
DevInfo::InfoRegisterTolocShell	261
plc::Interface	262
DevTc::EpicsInterface	248
InfoPlc::InfoInterface	256
TcComms::TCatInterface	307
EpicsTpy::macro_info	265
EpicsTpy::macro_record	265
ParseUtil::memory_location	266
ParseTpy::symbol_record	299
EpicsTpy::multi_io_support	270
EpicsTpy::epics_macrofiles_processing	234
ParseUtil::opc_list	274
ParseUtil::optarg	275
ParseTpy::parserinfo_type	278
ParseUtil::process_arg	281
DevTc::register_devsup	285
DevInfo::register_info_devsup	287
EpicsTpy::replacement_rules	288
EpicsTpy::epics_conversion	207
EpicsTpy::epics_db_processing	213
DevInfo::epics_info_db_processing	225
DevTc::epics_tc_db_processing	243
EpicsTpy::epics_list_processing	227
EpicsTpy::epics_macrofiles_processing	234
plc::scanner_thread_args	289
EpicsTpy::split_io_support	292
EpicsTpy::epics_db_processing	213
EpicsTpy::epics_list_processing	227
InfoPlc::stat_value	298
syminfo_processing	300
plc::System	300
ParseUtil::tag_processing	303
ParseTpy::tpy_file	323
TcComms::tcProcWrite	317
DevTc::tcRegisterTolocShell	321
DevTc::epics_record_traits< RecType >::traits_type	333
type_multipmap	333
ParseTpy::type_map	333
ParseUtil::variable_name	336

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ParseTpy::ads_routing_info	ADS routing information	113
TcComms::AmsRouterNotification	AMS Router Notification	116
std::atomic< string >	Atomic<string>	117
std::atomic< wstring >	Atomic<wstring>	118
std::atomic_string< stringT >	Atomic strings	120
ParseTpy::base_record	Base record definition	122
InfoPlc::BaseInfoItem	Base info item	125
plc::BasePLC	Base PLC	128
plc::BaseRecord	Class for managing a tag/channel	137
ParseUtil::bit_location	Bit location	157
std::case_char_traits	Case insensitive traits	158
std::case_wchar_traits	Case insensitive unicode traits	161
ParseTpy::compiler_info	Compiler information	163
TcComms::DataPar	Memory location struct	165
plc::DataValue	Data value	166
plc::DataValueTraits< T >	Data value traits	198
plc::DataValueTypeDef	Collection of type definitions	199
DevTc::devTcDefIn< RecType >	Device support input record	201

DevTc::devTcDeflo< RecType >	203
Device support record	
DevTc::devTcDefOut< RecType >	205
Device support output record	
DevTc::devTcDefWaveformIn< RecType >	206
Device support waveform record	
EpicsTpy::epics_conversion	207
Epics conversion	
EpicsTpy::epics_db_processing	213
Pics database record processing	
DevInfo::epics_info_db_processing	225
EPICS/Info db processing	
EpicsTpy::epics_list_processing	227
List processing	
EpicsTpy::epics_macrofiles_processing	234
Macro file processing	
DevTc::epics_record_traits< RecType >	242
Epics record traits	
DevTc::epics_tc_db_processing	243
EPICS/TCat db processing	
DevTc::EpicsInterface	248
Epics interface class	
std::std::hash< std::stringcase >	251
Hash for case insensitive string	
std::std::hash< std::wstringcase >	252
Hash for case insensitive unicode string	
InfoPlc::HistogramInfoItem	253
Histogram statistic	
InfoPlc::HistoryInfoItem	254
History tracker	
InfoPlc::InfoInterface	256
Info interface	
InfoPlc::InfoPLC	258
Info plc	
DevInfo::InfoRegisterToLocShell	261
Register info commands	
plc::Interface	262
Abstract interface	
ParseTpy::item_record	263
Item record	
EpicsTpy::macro_info	265
Macro information	
EpicsTpy::macro_record	265
Macro record	
ParseUtil::memory_location	266
Memory location	
EpicsTpy::multi_io_support	270
Multiple IO support	
ParseUtil::opc_list	274
OPC list	
ParseUtil::optarg	275
Optional arguments	
ParseTpy::parserinfo_type	278
ParseUtil::process_arg	281
Arguments for processing	
ParseTpy::project_record	284
Project information	

DevTc::register_devsup	Device support registration	285
DevInfo::register_info_devsup	Epics interface class	287
EpicsTpy::replacement_rules	Replacement rules	288
plc::scanner_thread_args	289
InfoPlc::SimpleInfoltem	Simple info tracker	290
EpicsTpy::split_io_support	Split IO support	292
InfoPlc::stat_value	Statistic value	298
ParseTpy::symbol_record	Symbol record	299
syminfo_processing	300
plc::System	System	300
ParseUtil::tag_processing	Tag processing selection	303
TcComms::TCatInterface	TCat interface class	307
TcComms::TcPLC	TwinCAT PLC	311
TcComms::tcProcWrite	TwinCAT process write requests	317
DevTc::tcRegisterTolocShell	Register TC commands	321
InfoPlc::TimeInfoltem	Timing tracker	321
ParseTpy::tpy_file	Tpy file parsing	323
DevTc::epics_record_traits< RecType >::traits_type	Epics record type	333
ParseTpy::type_map	Type dictionary	333
ParseTpy::type_record	334
ParseUtil::variable_name	Variable name	336

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

atomic_string.h	339
devInfo.cpp	340
devInfo.h	342
devTc.cpp	343
devTc.h	344
drvInfo.cpp	347
drvInfo.h	348
drvTc.cpp	349
drvTc.h	350
EpicsDbGen.cpp	351
infoPlc.cpp	353
infoPlc.h	354
infoPlcTemplate.h	356
iocMain.cpp	356
ParseTpy.cpp	358
ParseTpy.h	359
ParseTpyConst.h	360
ParseTpyInfo.cpp	363
ParseTpyTemplate.h	364
ParseUtil.cpp	365
ParseUtil.h	365
ParseUtilConst.h	367
plcBase.cpp	369
plcBase.h	370
plcBaseTemplate.h	372
stdafx.cpp	373
stdafx.h	373
stringcase.h	374
stringcase_hash.h	375
tcComms.cpp	376
tcComms.h	377
TpyToEpics.cpp	379
TpyToEpics.h	379
TpyToEpicsConst.h	381

Chapter 6

Module Documentation

6.1 Device support for TwinCAT/ADS

Classes

- class `DevTc::EpicsInterface`
Epics interface class.
- struct `DevTc::epics_record_traits< RecType >::traits_type`
Epics record type.
- struct `DevTc::epics_record_traits< RecType >`
Epics record traits.
- struct `DevTc::devTcDeflo< RecType >`
Device support record.
- struct `DevTc::devTcDefIn< RecType >`
Device support input record.
- struct `DevTc::devTcDefOut< RecType >`
device support output record.
- struct `DevTc::devTcDefWaveformIn< RecType >`
device support waveform record.

Typedefs

- typedef auto `DevTc::register_devsup::link_func(dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`
-> bool
Type descriping the link function.
- typedef std::pair< std::regex, link_func & > `DevTc::register_devsup::test_pattern`
pair of pattern and link function
- typedef std::vector< test_pattern > `DevTc::register_devsup::test_pattern_list`
list of pattern/link functions
- typedef epicsFloat64 `DevTc::epics_record_traits< RecType >::value_type`
Value type of (raw) value field.
- typedef `epics_record_traits< RecType >::traits_type DevTc::devTcDeflo< RecType >::rec_type`
Record type: aiRecord, etc.
- typedef `rec_type * DevTc::devTcDeflo< RecType >::rec_type_ptr`
Pointer to record type.

Enumerations

- enum `DevTc::epics_record_enum` {
 `DevTc::aaival` = 0, `DevTc::aaoval`, `DevTc::aival`, `DevTc::aoval`,
 `DevTc::bival`, `DevTc::boval`, `DevTc::eventval`, `DevTc::histogramval`,
 `DevTc::longinval`, `DevTc::longoutval`, `DevTc::mbbival`, `DevTc::mbboval`,
 `DevTc::mbbiDirectval`, `DevTc::mbboDirectval`, `DevTc::stringinval`, `DevTc::stringoutval`,
 `DevTc::waveformval`, `DevTc::airval`, `DevTc::aorval`, `DevTc::birval`,
 `DevTc::borval`, `DevTc::mbbiDirectval`, `DevTc::mbboDirectval`, `DevTc::mbbirval`,
 `DevTc::mbborval`, `DevTc::epics_record_enumEnd`, `DevTc::invalidval` = -1 }

Epics record type enum.

Functions

- static void `DevTc::register_devsup::add` (const std::regex &rgx, `link_func` &func)
Register a pattern/link function.
- static bool `DevTc::register_devsup::linkRecord` (const `std::stringcase` &inpout, dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)
linkRecord
- `DevTc::register_devsup::register_devsup` ()
Default constructor; adds a linkTcRecord entry.
- `DevTc::register_devsup::register_devsup` (const `register_devsup` &)
Disabled copy constructor.
- `register_devsup` & `DevTc::register_devsup::operator=` (const `register_devsup` &)
Disabled assignment operator.
- `DevTc::EpicsInterface::EpicsInterface` (plc::BaseRecord &dval)
Constructor.
- `DevTc::EpicsInterface::~EpicsInterface` ()
Deconstructor.
- void `DevTc::EpicsInterface::set_isPassive` (bool passive)
Set isPassive.
- bool `DevTc::EpicsInterface::get_isCallback` () const
Get isCallback.
- void `DevTc::EpicsInterface::set_isCallback` (bool isCb)
Set isCallback.
- void `DevTc::EpicsInterface::set_pEpicsRecord` (dbCommon *pEpRecord)
Set pEpicsRecord.
- void * `DevTc::EpicsInterface::get_pEpicsVal` () const
Get pEpicsVal.
- void `DevTc::EpicsInterface::set_pEpicsVal` (void *pVal)
Set pEpicsVal.
- unsigned long `DevTc::EpicsInterface::get_size` () const
Get size.
- void `DevTc::EpicsInterface::set_size` (unsigned long nBytes)
Set size.
- bool `DevTc::EpicsInterface::get_callbackRequestPending` () const
Get callbackRequestPending.
- const CALLBACK & `DevTc::EpicsInterface::callback` () const
Get pointer to callback structure.
- CALLBACK & `DevTc::EpicsInterface::callback` ()
Get pointer to callback structure.

- const IOSCANPVT & `DevTc::EpicsInterface::ioscan () const`
Get reference to io scan list pointer.
- IOSCANPVT & `DevTc::EpicsInterface::ioscan ()`
Get reference to io scan list pointer.
- IOSCANPVT `DevTc::EpicsInterface::get_ioscan () const`
Get pointer to io scan list.
- void `DevTc::EpicsInterface::set_ioscan (const IOSCANPVT ioscan)`
Set pointer to io scan list.
- virtual bool `DevTc::EpicsInterface::push () override`
Makes a call to the EPICS dbProcess function.
- virtual bool `DevTc::EpicsInterface::pull () override`
Does nothing.
- static const char *const `DevTc::epics_record_traits< RecType >::name ()`
Name of the record.
- static `value_type * DevTc::epics_record_traits< RecType >::val (traits_type *prec)`
Returns the (raw) value of a record.
- static bool `DevTc::epics_record_traits< RecType >::read (traits_type *epicsrec, plc::BaseRecord *baserec)`
Performs the read access on prec.
- static bool `DevTc::epics_record_traits< RecType >::write (plc::BaseRecord *baserec, traits_type *epicsrec)`
Performs the write access on prec.
- `DevTc::devTcDeflo< RecType >::devTcDeflo ()`
Hide constructor.
- static long `DevTc::devTcDeflo< RecType >::get_ioint_info (int cmd, dbCommon *prec, IOSCANPVT *ppvt)`
IO/INT info callback.
- `DevTc::devTcDefIn< RecType >::devTcDefIn ()`
Constructor.
- static long `DevTc::devTcDefIn< RecType >::init_read_record (rec_type_ptr prec)`
init callback for read records
- static long `DevTc::devTcDefIn< RecType >::read (rec_type_ptr precord)`
read callback
- `DevTc::devTcDefOut< RecType >::devTcDefOut ()`
Constructor.
- static long `DevTc::devTcDefOut< RecType >::init_write_record (rec_type_ptr prec)`
init callback for write records
- static long `DevTc::devTcDefOut< RecType >::write (rec_type_ptr precord)`
write callback
- `DevTc::devTcDefWaveformIn< RecType >::devTcDefWaveformIn ()`
Constructor.
- static long `DevTc::devTcDefWaveformIn< RecType >::init_read_waveform_record (rec_type_ptr prec)`
init callback for read records
- static long `DevTc::devTcDefWaveformIn< RecType >::read_waveform (rec_type_ptr precord)`
read callback

Variables

- `test_pattern_list DevTc::register_devsup::tp_list`
list of pattern and links
- static `register_devsup DevTc::register_devsup::the_register_devsup`
the one global instance of the register class
- bool `DevTc::EpicsInterface::isPassive`

- bool `DevTc::EpicsInterface::isCallback`
- dbCommon * `DevTc::EpicsInterface::pEpicsRecord`
Pointer to the EPICS record.
- void * `DevTc::EpicsInterface::pEpicsVal`
Pointer to the RVAL field of the EPICS record.
- unsigned long `DevTc::EpicsInterface::size`
Size (bytes) of the data.
- IOSCANPVT `DevTc::EpicsInterface::ioscanpvt`
Pointer to IO scan list.
- CALLBACK `DevTc::EpicsInterface::callbackval`
Callback structure.
- double `DevTc::epics_record_traits< RecType >::traits_type::val`
Value.
- static const aitEnum `DevTc::epics_record_traits< RecType >::value_ait_type` = aitEnumFloat64
- static const aitInt32 `DevTc::epics_record_traits< RecType >::value_count` = 0
- static const int `DevTc::epics_record_traits< RecType >::value_conversion` = 0
return value for read_io functions 0=default, 2=don't convert
- static const bool `DevTc::epics_record_traits< RecType >::input_record` = true
Indicates if this is an input record.
- static const bool `DevTc::epics_record_traits< RecType >::raw_record` = false
Indicates if this is a raw record.
- long `DevTc::devTcDeflo< RecType >::number`
Number of support functions.
- DEVSUPFUN `DevTc::devTcDeflo< RecType >::report_fn`
Report support function.
- DEVSUPFUN `DevTc::devTcDeflo< RecType >::init_fn`
Init support function.
- DEVSUPFUN `DevTc::devTcDeflo< RecType >::init_record_fn`
Record init support function.
- DEVSUPFUN `DevTc::devTcDeflo< RecType >::get_ioint_info_fn`
IO/INT support function.
- DEVSUPFUN `DevTc::devTcDeflo< RecType >::io_fn`
Read/write support function.
- DEVSUPFUN `DevTc::devTcDeflo< RecType >::special_linconv_fn`
Linear conversion support function.

6.1.1 Detailed Description

6.1.2 Enumeration Type Documentation

6.1.2.1 epics_record_enum

```
enum DevTc::epics_record_enum
```

Epics record type enum.

This record type enums are used as index the epics traits class

Enumerator

aaival	double input array
aoaval	double output array
aival	double input
aoval	double output
bival	binary input
boval	binary output
eventval	event
histogramval	histogram
longinval	integer input
longoutval	integer output
mbbival	enum input
mbboval	enum output
mbbiDirectval	enum input direct
mbboDirectval	enum output direct
stringinval	string input
stringoutval	string output
waveformval	waveform
airval	raw double input
aorval	raw double output
birval	raw binary input
borval	raw binary output
mbbiDirectrvl	raw enum input
mbboDirectrvl	raw enum output
mbbirval	raw enum input direct
mbborval	raw enum output direct
epics_record_enumEnd	End of enum (sentinel value)
invalidval	invalid

Definition at line 197 of file devTc.h.

6.1.3 Function Documentation

6.1.3.1 EpicsInterface()

```
DevTc::EpicsInterface::EpicsInterface (
    plc::BaseRecord & dval )
```

Constructor.

[EpicsInterface::EpicsInterface](#)

Definition at line 153 of file devTc.cpp.

References DevTc::EpicsInterface::callbackval.

6.1.3.2 get_callbackRequestPending()

```
bool DevTc::EpicsInterface::get_callbackRequestPending ( ) const
```

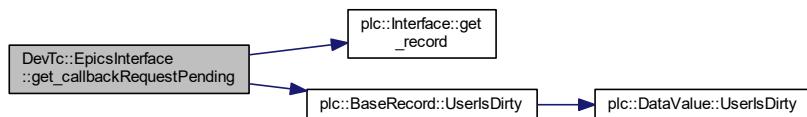
Get callbackRequestPending.

[EpicsInterface::get_callbackRequestPending](#)

Definition at line 164 of file devTc.cpp.

References plc::Interface::get_record(), and plc::BaseRecord::UserIsDirty().

Here is the call graph for this function:



6.1.3.3 linkRecord()

```
bool DevTc::register_devsup::linkRecord (
    const std::string& inout,
    dbCommon * pEpicsRecord,
    plc::BaseRecordPtr & pRecord ) [static]
```

linkRecord

Go through list and call first link function which matches the pattern Used to link epics records with internal records.

Parameters

<i>inout</i>	Value of INP/OUT field
<i>pEpicsRecord</i>	Pointer to EPICS record
<i>pRecord</i>	Pointer to a base record (return)

Returns

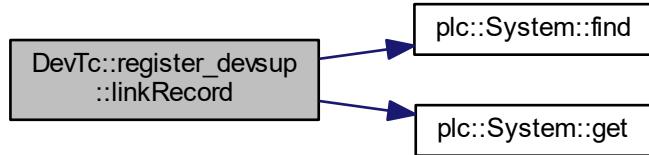
true if one match was found and successfully linked

[register_devsup::linkRecord](#)

Definition at line 114 of file devTc.cpp.

References plc::System::find(), and plc::System::get().

Here is the call graph for this function:



6.1.3.4 push()

```
bool DevTc::EpicsInterface::push( ) [override], [virtual]
```

Makes a call to the EPICS dbProcess function.

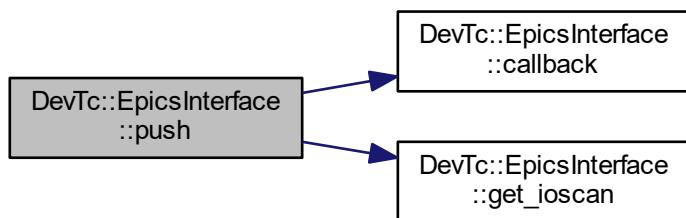
[EpicsInterface::push](#)

Implements [plc::Interface](#).

Definition at line 173 of file devTc.cpp.

References [DevTc::EpicsInterface::callback\(\)](#), [DevTc::EpicsInterface::get_ioscan\(\)](#), [DevTc::EpicsInterface::isCallback](#), and [DevTc::EpicsInterface::isPassive](#).

Here is the call graph for this function:



6.1.4 Variable Documentation

6.1.4.1 ioscanpvt

```
IOSCANPVT DevTc::EpicsInterface::ioscanpvt [protected]
```

Pointer to IO scan list.

Bool indicating that a read callback is pending Set to true for in/out records when callback request is made. true : if this is an in/out record, then dbProcess will do an EPICS read instead of write, then reset this value to false

Definition at line 188 of file devTc.h.

Referenced by DevTc::EpicsInterface::get_ioscan(), DevTc::EpicsInterface::ioscan(), and DevTc::EpicsInterface::set_ioscan().

6.1.4.2 isCallback

```
bool DevTc::EpicsInterface::isCallback [protected]
```

Bool indicating whether callback is needed to call dbProcess true : SCAN = I/O Intr or the record is an out record

Definition at line 175 of file devTc.h.

Referenced by DevTc::EpicsInterface::get_isCallback(), DevTc::EpicsInterface::push(), and DevTc::EpicsInterface::set_isCallback().

6.1.4.3 isPassive

```
bool DevTc::EpicsInterface::isPassive [protected]
```

Bool indicating passive scan true : EPICS record SCAN field is set to PASSIVE

Definition at line 172 of file devTc.h.

Referenced by DevTc::EpicsInterface::push(), and DevTc::EpicsInterface::set_isPassive().

6.1.4.4 the_register_devsup

```
register_devsup DevTc::register_devsup::the_register_devsup [static], [protected]
```

the one global instance of the register class

[register_devsup::the_register_devsup](#)

Definition at line 100 of file devTc.h.

Referenced by DevTc::register_devsup::add().

6.1.4.5 value_ait_type

```
template<epics_record_enum RecType>
const aitEnum DevTc::epics_record_traits< RecType >::value_ait_type = aitEnumFloat64 [static]
```

Data type of records val/rval field aitEnumInvalid type signals an array. Take type/len from record

Definition at line 272 of file devTc.h.

6.1.4.6 value_count

```
template<epics_record_enum RecType>
const aitInt32 DevTc::epics_record_traits< RecType >::value_count = 0 [static]
```

Array length: 1=scalar value, 0=array - see nelm for length, fixed for strings according to the record

Definition at line 275 of file devTc.h.

6.2 Functions called by the EPICS base

Functions

- void [DevInfo::infoLoadRecords](#) (const iocshArgBuf *args)
Info load records.
- void [DevInfo::infoSetScanRate](#) (const iocshArgBuf *args)
Info set scan rate.
- void [DevInfo::infoList](#) (const iocshArgBuf *args)
channel lists
- void [DevInfo::infoAlias](#) (const iocshArgBuf *args)
alias
- void [DevInfo::infoPrefix](#) (const iocshArgBuf *args)
tag prefix
- void [DevInfo::infoPrintVals](#) (const iocshArgBuf *args)
Info print vals.
- void [DevTc::tcLoadRecords](#) (const iocshArgBuf *args)
TCat load records.
- void [DevTc::tcSetScanRate](#) (const iocshArgBuf *args)
TCat set scan rate.
- void [DevTc::tcList](#) (const iocshArgBuf *args)
channel lists
- void [DevTc::tcMacro](#) (const iocshArgBuf *args)
macro files
- void [DevTc::tcAlias](#) (const iocshArgBuf *args)
alias
- void [DevTc::tcPrintVals](#) (const iocshArgBuf *args)
TCat print vals.

6.2.1 Detailed Description

6.2.2 Function Documentation

6.2.2.1 infoAlias()

```
void DevInfo::infoAlias (
    const iocshArgBuf * args )
```

alias

Define a nick name or alias

Definition at line 346 of file drvInfo.cpp.

References [plc::System::get\(\)](#).

Here is the call graph for this function:



6.2.2.2 infoList()

```
void DevInfo::infoList (
    const iocshArgBuf * args )
```

channel lists

List function to generate separate listings

Definition at line 322 of file drvInfo.cpp.

References plc::System::get().

Here is the call graph for this function:



6.2.2.3 infoLoadRecords()

```
void DevInfo::infoLoadRecords (
    const iocshArgBuf * args )
```

Info load records.

Function for loading the info record, and using it to generate internal record entries as well as the EPICS .db file

Definition at line 166 of file drvInfo.cpp.

6.2.2.4 infoPrefix()

```
void DevInfo::infoPrefix (
    const iocshArgBuf * args )
```

tag prefix

Define the tag prefix

Definition at line 369 of file drvInfo.cpp.

References plc::System::get().

Here is the call graph for this function:



6.2.2.5 infoPrintVals()

```
void DevInfo::infoPrintVals (
    const iocshArgBuf * args )
```

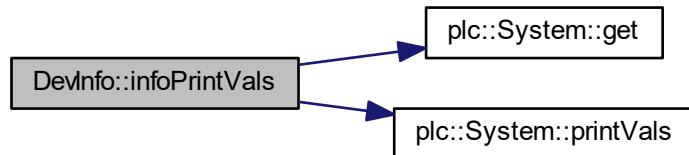
Info print vals.

Debugging function that prints the values for all records on the PLCs

Definition at line 392 of file drvInfo.cpp.

References plc::System::get(), and plc::System::printVals().

Here is the call graph for this function:



6.2.2.6 infoSetScanRate()

```
void DevInfo::infoSetScanRate (
    const iocshArgBuf * args )
```

Info set scan rate.

Set scan rate of the read scanner

Definition at line 281 of file drvInfo.cpp.

6.2.2.7 tcAlias()

```
void DevTc::tcAlias (
    const iocshArgBuf * args )
```

alias

Define a nick name or alias

Definition at line 651 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



6.2.2.8 tcList()

```
void DevTc::tcList (
    const iocshArgBuf * args )
```

channel lists

List function to generate separate listings

Definition at line 603 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



6.2.2.9 tcLoadRecords()

```
void DevTc::tcLoadRecords (
    const iocshArgBuf * args )
```

TCat load records.

Function for loading a TCat tpy file, and using it to generate internal record entries as well as the EPICs .db file

Definition at line 390 of file drvTc.cpp.

6.2.2.10 tcMacro()

```
void DevTc::tcMacro (
    const iocshArgBuf * args )
```

macro files

Macro function to generate macro files

Definition at line 627 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



6.2.2.11 tcPrintVals()

```
void DevTc::tcPrintVals (
    const iocshArgBuf * args )
```

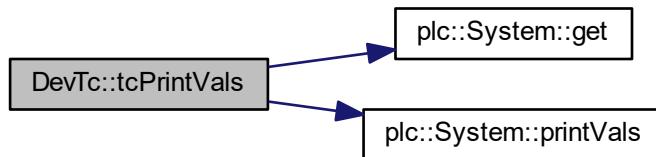
TCat print vals.

Debugging function that prints the values for all records on the PLCs

Definition at line 703 of file drvTc.cpp.

References plc::System::get(), and plc::System::printVals().

Here is the call graph for this function:



6.2.2.12 tcSetScanRate()

```
void DevTc::tcSetScanRate (
    const iocshArgBuf * args )
```

TCat set scan rate.

Set scan rate of the read scanner

Definition at line 547 of file drvTc.cpp.

References plc::System::get().

Here is the call graph for this function:



6.3 Constants related to read/write scanning

Variables

- const int `InfoPlc::default_scanrate` = 100
 - default PLC TwinCAT scan rate (100ms)*
- const int `InfoPlc::minimum_scanrate` = 5
 - minimum PLC TwinCAT scan rate (5ms)*
- const int `InfoPlc::maximum_scanrate` = 10000
 - maximum PLC TwinCAT scan rate (10s)*
- const int `TcComms::MAX_REQ_SIZE` = 250000
 - maximum allowed request size (bytes)*
- const int `TcComms::MAX_SINGLE_GAP_SIZE` = 50
 - maximum allowed size (bytes) of a memory gap within continuous request*
- const double `TcComms::MAX_REL_GAP` = 0.25
 - (maximum allowed total gap size) / (current request size)*
- const int `TcComms::MIN_REL_GAP_SIZE` = 100
 - minimum allowed relative gap size (bytes)*
- const int `TcComms::default_scanrate` = 100
 - default PLC TwinCAT scan rate (100ms)*
- const int `TcComms::minimum_scanrate` = 5
 - minimum PLC TwinCAT scan rate (5ms)*
- const int `TcComms::maximum_scanrate` = 10000
 - maximum PLC TwinCAT scan rate (10s)*
- const int `TcComms::default_multiple` = 10
 - default multiple for PLC EPICS scan rate (10)*
- const int `TcComms::minimum_multiple` = 1
 - minimum multiple for PLC EPICS scan rate (1)*
- const int `TcComms::maximum_multiple` = 200
 - maximum multiple for PLC EPICS scan rate (200)*

6.3.1 Detailed Description

6.4 Classes for describing project information

Classes

- class [ParseTpy::ads_routing_info](#)
ADS routing information.
- class [ParseTpy::compiler_info](#)
Compiler information.
- class [ParseTpy::project_record](#)
Project information.

6.4.1 Detailed Description

6.5 Classes for describing a structure element

Classes

- class `ParseTpy::base_record`
Base record definition.
- class `ParseTpy::item_record`
item record

Typedefs

- `typedef std::pair< int, int > ParseTpy::dimension`
- `typedef std::list< dimension > ParseTpy::dimensions`
- `typedef std::map< int, std::stringcase > ParseTpy::enum_map`
- `typedef std::pair< int, std::stringcase > ParseTpy::enum_pair`
- `typedef std::list< item_record > ParseTpy::item_list`

6.5.1 Detailed Description

6.5.2 Typedef Documentation

6.5.2.1 dimension

```
typedef std::pair<int, int> ParseTpy::dimension
```

This class stores a lbound, elements pair.

Definition at line 217 of file ParseTpy.h.

6.5.2.2 dimensions

```
typedef std::list<dimension> ParseTpy::dimensions
```

This list stores lbound, elements pairs.

Definition at line 221 of file ParseTpy.h.

6.5.2.3 enum_map

```
typedef std::map<int, std::stringcase> ParseTyp::enum_map
```

This map stores a list of enum values.

Definition at line 225 of file ParseTyp.h.

6.5.2.4 enum_pair

```
typedef std::pair<int, std::stringcase> ParseTyp::enum_pair
```

This type stores an enum pair.

Definition at line 229 of file ParseTyp.h.

6.5.2.5 item_list

```
typedef std::list<item_record> ParseTyp::item_list
```

This class stores a list of subitems.

Definition at line 243 of file ParseTyp.h.

6.6 Classes for describing a type

Classes

- class `ParseTpy::type_record`
- class `ParseTpy::type_map`

Type dictionary.

Typedefs

- `typedef std::multimap< unsigned int, type_record > ParseTpy::type_multipmap`

Enumerations

- enum `ParseTpy::type_enum` {
 `ParseTpy::unknown, ParseTpy::simple, ParseTpy::arraytype, ParseTpy::enumtype,`
 `ParseTpy::structtype, ParseTpy::functionblock` }

Type enum.

6.6.1 Detailed Description

6.6.2 Typedef Documentation

6.6.2.1 type_multipmap

```
typedef std::multimap<unsigned int, type_record> ParseTpy::type_multipmap
```

This is a multimap to store type records

Definition at line 316 of file ParseTpy.h.

6.6.3 Enumeration Type Documentation

6.6.3.1 type_enum

```
enum ParseTpy::type_enum
```

Type enum.

This structure describes a type record

Enumerator

unknown	Unknown type.
simple	Simple type.
arraytype	Array type.
enumtype	Enumerated type.
structtype	Structure type.
functionblock	Function block.

Definition at line 254 of file ParseTyp.h.

6.7 Classes for describing a symbol

Classes

- class [ParseTpy::symbol_record](#)

Symbol record.

TypeDefs

- [typedef std::list< symbol_record > ParseTpy::symbol_list](#)

6.7.1 Detailed Description

6.7.2 Typedef Documentation

6.7.2.1 symbol_list

```
typedef std::list<symbol\_record> ParseTpy::symbol\_list
```

This is a list of symbol records

Definition at line 359 of file [ParseTpy.h](#).

6.8 Classes for describing the parser

Classes

- class [ParseTpy::tpy_file](#)

Tpy file parsing.

6.8.1 Detailed Description

6.9 XML tpy file constants

Variables

- const char *const ParseTpy::xmlPlcProjectInfo = "PlcProjectInfo"
PLC project info.
- const char *const ParseTpy::xmlProjectInfo = "ProjectInfo"
Project info.
- const char *const ParseTpy::xmlRoutingInfo = "RoutingInfo"
Routing info.
- const char *const ParseTpy::xmlCompilerInfo = "CompilerInfo"
Compiler info.
- const char *const ParseTpy::xmlAdsInfo = "AdsInfo"
ADS info.
- const char *const ParseTpy::xmlDataTypes = "DataTypes"
Data types.
- const char *const ParseTpy::xmlDataType = "DataType"
Data type.
- const char *const ParseTpy::xmlSymbols = "Symbols"
Symbols.
- const char *const ParseTpy::xmlSymbol = "Symbol"
Symbol.
- const char *const ParseTpy::xmlProperties = "Properties"
Properties.
- const char *const ParseTpy::xmlProperty = "Property"
Property.
- const char *const ParseTpy::xmlCompilerVersion = "CompilerVersion"
Compiler version.
- const char *const ParseTpy::xmlTwinCATVersion = "TwinCATVersion"
TwinCAT version.
- const char *const ParseTpy::xmlCpuFamily = "CpuFamily"
CPU family.
- const char *const ParseTpy::xmlNetId = "NetId"
Net ID.
- const char *const ParseTpy::xmlPort = "Port"
Port.
- const char *const ParseTpy::xmlTargetName = "TargetName"
Target name.
- const char *const ParseTpy::xmlName = "Name"
Name.
- const char *const ParseTpy::xmlType = "Type"
Type.
- const char *const ParseTpy::xmlAttrDecoration = "Decoration"
Decoration.
- const char *const ParseTpy::xmlAttrPointer = "Pointer"
Pointer.
- const char *const ParseTpy::xmlIGroup = "IGroup"
I Group.
- const char *const ParseTpy::xmlIOffset = "IOffset"
I Offset.

- const char *const ParseTpy::xmlBitSize = "BitSize"
Bit size.
- const char *const ParseTpy::xmlBitOffs = "BitOffs"
Bit Offset.
- const char *const ParseTpy::xmlArrayInfo = "ArrayInfo"
Array info.
- const char *const ParseTpy::xmlArrayLBound = "LBound"
Lower bound.
- const char *const ParseTpy::xmlArrayElements = "Elements"
Elements.
- const char *const ParseTpy::xmlSubItem = "SubItem"
Sub item.
- const char *const ParseTpy::xmlFbInfo = "FbInfo"
Fb info.
- const char *const ParseTpy::xmlEnumInfo = "EnumInfo"
Enum info.
- const char *const ParseTpy::xmlEnumText = "Text"
Text.
- const char *const ParseTpy::xmlEnumEnum = "Enum"
Enum.
- const char *const ParseTpy::xmlEnumComment = "Comment"
Comment.
- const char *const ParseTpy::xmlValue = "Value"
Value.
- const char *const ParseTpy::xmlDesc = "Desc"
Description.

6.9.1 Detailed Description

6.10 OPC tpy file constants

Variables

- const char *const ParseTpy::opcExport = "opc"
OPC.
- const char *const ParseTpy::opcProp = "opc_prop"
OPC property.
- const char *const ParseTpy::opcBracket = "["
OPC bracket.

6.10.1 Detailed Description

6.11 Option processing

Classes

- class [ParseUtil::optarg](#)

Optional arguments.

6.11.1 Detailed Description

These function and classes are used to parse command line options

6.12 OPC related functions and classes

Classes

- class `ParseUtil::opc_list`

OPC list.

Typedefs

- `typedef std::map< int, std::stringcase > ParseUtil::property_map`
- `typedef std::pair< int, std::stringcase > ParseUtil::property_el`

Enumerations

- enum `ParseUtil::opc_enum { ParseUtil::no_change, ParseUtil::publish, ParseUtil::silent }`

OPC state enum.

6.12.1 Detailed Description

These function and classes are generally used to describe channel properties

6.12.2 Typedef Documentation

6.12.2.1 `property_el`

```
typedef std::pair<int, std::stringcase> ParseUtil::property_el
```

This pair stores one element of opc properties.

Definition at line 95 of file ParseUtil.h.

6.12.2.2 `property_map`

```
typedef std::map<int, std::stringcase> ParseUtil::property_map
```

This map stores a list of opc properties.

Definition at line 91 of file ParseUtil.h.

6.12.3 Enumeration Type Documentation

6.12.3.1 `opc_enum`

```
enum ParseUtil::opc_enum
```

OPC state enum.

This enum denotes the opc state.

Enumerator

no_change	Do not change inherited behaviour.
publish	Publish.
silent	Do not publish.

Definition at line 79 of file ParseUtil.h.

6.13 Classes for describing a parser argument

Classes

- class [ParseUtil::variable_name](#)
Variable name.
- class [ParseUtil::bit_location](#)
Bit location.
- class [ParseUtil::memory_location](#)
Memory location.
- class [ParseUtil::process_arg](#)
Arguments for processing.
- class [ParseUtil::tag_processing](#)
Tag processing selection.

Enumerations

- enum [ParseUtil::process_type_enum](#) {
 ParseUtil::pt_invalid, ParseUtil::pt_int, ParseUtil::pt_real, ParseUtil::pt_bool,
 ParseUtil::pt_string, ParseUtil::pt_enum, ParseUtil::pt_binary }
Process type.
- enum [ParseUtil::process_tag_enum](#) { [ParseUtil::process_all](#), [ParseUtil::process_atomic](#), [ParseUtil::process_structured](#) }
Tag preoicessing enum.

6.13.1 Detailed Description

6.13.2 Enumeration Type Documentation

6.13.2.1 process_tag_enum

enum [ParseUtil::process_tag_enum](#)

Tag preoicessing enum.

Enumerated type to describe the tag processing

Enumerator

process_all	Process all data types.
process_atomic	Process atomic data types.
process_structured	Process structured data type (array, struct, function block)

Definition at line 365 of file ParseUtil.h.

6.13.2.2 process_type_enum

```
enum ParseUtil::process_type_enum
```

Process type.

Enumerated type to describe the process type

Enumerator

pt_invalid	Invalid type.
pt_int	Numeral type.
pt_real	Floating point type.
pt_bool	Logic type.
pt_string	String type.
pt_enum	Enumerated type.
pt_binary	Binary type.

Definition at line 278 of file ParseUtil.h.

6.14 OPC property constants

Variables

- const int ParseUtil::OPC_PROP_CDT = 1
- const int ParseUtil::OPC_PROP_VALUE = 2
- const int ParseUtil::OPC_PROP_QUALITY = 3
- const int ParseUtil::OPC_PROP_TIME = 4
- const int ParseUtil::OPC_PROP_RIGHTS = 5
- const int ParseUtil::OPC_PROP_SCANRATE = 6
- const int ParseUtil::OPC_PROP_UNIT = 100
- const int ParseUtil::OPC_PROP_DESC = 101
- const int ParseUtil::OPC_PROP_HIEU = 102
- const int ParseUtil::OPC_PROP_LOEU = 103
- const int ParseUtil::OPC_PROP_HIRANGE = 104
- const int ParseUtil::OPC_PROP_LORANGE = 105
- const int ParseUtil::OPC_PROP_CLOSE = 106
- const int ParseUtil::OPC_PROP_OPEN = 107
- const int ParseUtil::OPC_PROP_TIMEZONE = 108
- const int ParseUtil::OPC_PROP_FGC = 201
- const int ParseUtil::OPC_PROP_BGC = 202
- const int ParseUtil::OPC_PROP_BLINK = 203
- const int ParseUtil::OPC_PROP_BMP = 204
- const int ParseUtil::OPC_PROP SND = 205
- const int ParseUtil::OPC_PROP_HTML = 206
- const int ParseUtil::OPC_PROP_AVI = 207
- const int ParseUtil::OPC_PROP_ALMSTAT = 300
- const int ParseUtil::OPC_PROP_ALMHELP = 301
- const int ParseUtil::OPC_PROP_ALMAREAS = 302
- const int ParseUtil::OPC_PROP_ALMPRIMARYAREA = 303
- const int ParseUtil::OPC_PROP_ALMCCONDITION = 304
- const int ParseUtil::OPC_PROP_ALMLIMIT = 305
- const int ParseUtil::OPC_PROP_ALMDB = 306
- const int ParseUtil::OPC_PROP_ALMH = 307
- const int ParseUtil::OPC_PROP_ALMH = 308
- const int ParseUtil::OPC_PROP_ALML = 309
- const int ParseUtil::OPC_PROP_ALMLL = 310
- const int ParseUtil::OPC_PROP_ALMROC = 311
- const int ParseUtil::OPC_PROP_ALMDEV = 312
- const int ParseUtil::OPC_PROP_PREC = 8500
- const int ParseUtil::OPC_PROP_ZRST = 8510
- const int ParseUtil::OPC_PROP_FFST = 8525
- const int ParseUtil::OPC_PROP_RECTYPE = 8600
- const int ParseUtil::OPC_PROP_INOUT = 8601
- const char *const ParseUtil::OPC_PROP_INPUT = "input"
- const char *const ParseUtil::OPC_PROP_OUTPUT = "output"
- const int ParseUtil::OPC_PROP_TSE = 8602
- const int ParseUtil::OPC_PROP_PINI = 8603
- const int ParseUtil::OPC_PROP_DTYP = 8604
- const int ParseUtil::OPC_PROP_SERVER = 8610
- const int ParseUtil::OPC_PROP_PLNAME = 8611
- const int ParseUtil::OPC_PROP_ALIAS = 8620
- const int ParseUtil::OPC_PROP_ALMOSV = 8700
- const int ParseUtil::OPC_PROP_ALMZSV = 8701

- const int ParseUtil::OPC_PROP_ALMCOSV = 8702
- const int ParseUtil::OPC_PROP_ALMUNSV = 8703
- const int ParseUtil::OPC_PROP_ALMZRSV = 8710
- const int ParseUtil::OPC_PROP_ALMFFSV = 8725
- const int ParseUtil::OPC_PROP_ALMHHSV = 8727
- const int ParseUtil::OPC_PROP_ALMHHSV = 8728
- const int ParseUtil::OPC_PROP_ALMLSV = 8729
- const int ParseUtil::OPC_PROP_ALMLLSV = 8730

6.14.1 Detailed Description

6.14.2 Variable Documentation

6.14.2.1 OPC_PROP_ALIAS

const int ParseUtil::OPC_PROP_ALIAS = 8620

alias for structure item or symbol name

Definition at line 69 of file ParseUtilConst.h.

Referenced by ParseUtil::variable_name::append(), and EpicsTpy::epics_db_processing::operator()().

6.14.2.2 OPC_PROP_ALMAREAS

const int ParseUtil::OPC_PROP_ALMAREAS = 302

area

Definition at line 45 of file ParseUtilConst.h.

6.14.2.3 OPC_PROP_ALMCONDITION

const int ParseUtil::OPC_PROP_ALMCONDITION = 304

condition

Definition at line 47 of file ParseUtilConst.h.

6.14.2.4 OPC_PROP_ALMCOSV

```
const int ParseUtil::OPC_PROP_ALMCOSV = 8702
```

alarm: change of state severity

Definition at line 72 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.5 OPC_PROP_ALMDB

```
const int ParseUtil::OPC_PROP_ALMDB = 306
```

dead band, tolerance

Definition at line 49 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.6 OPC_PROP_ALMDEV

```
const int ParseUtil::OPC_PROP_ALMDEV = 312
```

deviation

Definition at line 55 of file ParseUtilConst.h.

6.14.2.7 OPC_PROP_ALMFFSV

```
const int ParseUtil::OPC_PROP_ALMFFSV = 8725
```

alarm: fifteen state severity

Definition at line 75 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.8 OPC_PROP_ALMH

```
const int ParseUtil::OPC_PROP_ALMH = 308
```

high alarm -> HIGH

Definition at line 51 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.9 OPC_PROP_ALMHELP

```
const int ParseUtil::OPC_PROP_ALMHELP = 301
```

help

Definition at line 44 of file ParseUtilConst.h.

6.14.2.10 OPC_PROP_ALMHH

```
const int ParseUtil::OPC_PROP_ALMHH = 307
```

high high alarm -> HIHI

Definition at line 50 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.11 OPC_PROP_ALMHHSV

```
const int ParseUtil::OPC_PROP_ALMHHSV = 8727
```

alarm: hihi severity

Definition at line 76 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.12 OPC_PROP_ALMHSV

```
const int ParseUtil::OPC_PROP_ALMHSV = 8728
```

alarm: high severity

Definition at line 77 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.13 OPC_PROP_ALML

```
const int ParseUtil::OPC_PROP_ALML = 309
```

low alarm -> LOW

Definition at line 52 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.14 OPC_PROP_ALMLIMIT

```
const int ParseUtil::OPC_PROP_ALMLIMIT = 305
```

limit

Definition at line 48 of file ParseUtilConst.h.

6.14.2.15 OPC_PROP_ALMLL

```
const int ParseUtil::OPC_PROP_ALMLL = 310
```

low low alaam -> LOLO

Definition at line 53 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.16 OPC_PROP_ALMLSV

```
const int ParseUtil::OPC_PROP_ALMLSV = 8730
```

alarm: lolo severity

Definition at line 79 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.17 OPC_PROP_ALMLSV

```
const int ParseUtil::OPC_PROP_ALMLSV = 8729
```

alarm: low severity

Definition at line 78 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.18 OPC_PROP_ALMOSV

```
const int ParseUtil::OPC_PROP_ALMOSV = 8700
```

alarm: one severity

Definition at line 70 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.19 OPC_PROP_ALMPRIMARYAREA

```
const int ParseUtil::OPC_PROP_ALMPRIMARYAREA = 303
```

primery area

Definition at line 46 of file ParseUtilConst.h.

6.14.2.20 OPC_PROP_ALMROC

```
const int ParseUtil::OPC_PROP_ALMROC = 311
```

rate of change

Definition at line 54 of file ParseUtilConst.h.

6.14.2.21 OPC_PROP_ALMSTAT

```
const int ParseUtil::OPC_PROP_ALMSTAT = 300  
status
```

Definition at line 43 of file ParseUtilConst.h.

6.14.2.22 OPC_PROP_ALMUNSV

```
const int ParseUtil::OPC_PROP_ALMUNSV = 8703  
alarm: unknown state severity
```

Definition at line 73 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.23 OPC_PROP_ALMZRSV

```
const int ParseUtil::OPC_PROP_ALMZRSV = 8710  
alarm: one state severity
```

Definition at line 74 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.24 OPC_PROP_ALMZSV

```
const int ParseUtil::OPC_PROP_ALMZSV = 8701  
alarm: zero severity
```

Definition at line 71 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.25 OPC_PROP_AVI

```
const int ParseUtil::OPC_PROP_AVI = 207  
avi file
```

Definition at line 41 of file ParseUtilConst.h.

6.14.2.26 OPC_PROP_BGC

```
const int ParseUtil::OPC_PROP_BGC = 202
```

background color

Definition at line 36 of file ParseUtilConst.h.

6.14.2.27 OPC_PROP_BLINK

```
const int ParseUtil::OPC_PROP_BLINK = 203
```

blinking

Definition at line 37 of file ParseUtilConst.h.

6.14.2.28 OPC_PROP_BMP

```
const int ParseUtil::OPC_PROP_BMP = 204
```

bmp file

Definition at line 38 of file ParseUtilConst.h.

6.14.2.29 OPC_PROP_CDT

```
const int ParseUtil::OPC_PROP_CDT = 1
```

canonocal data type

Definition at line 18 of file ParseUtilConst.h.

6.14.2.30 OPC_PROP_CLOSE

```
const int ParseUtil::OPC_PROP_CLOSE = 106
```

label for close state -> ONAM

Definition at line 31 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

6.14.2.31 OPC_PROP_DESC

```
const int ParseUtil::OPC_PROP_DESC = 101
```

description string -> DESC

Definition at line 26 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.32 OPC_PROP_DTYP

```
const int ParseUtil::OPC_PROP_DTYP = 8604
```

DTYP field: opc or opcRaw

Definition at line 66 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.33 OPC_PROP_FFST

```
const int ParseUtil::OPC_PROP_FFST = 8525
```

... fifteen string

Definition at line 59 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

6.14.2.34 OPC_PROP_FGC

```
const int ParseUtil::OPC_PROP_FGC = 201
```

foreground color

Definition at line 35 of file ParseUtilConst.h.

6.14.2.35 OPC_PROP_HIEU

```
const int ParseUtil::OPC_PROP_HIEU = 102
```

high expectation value -> HOPR

Definition at line 27 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.36 OPC_PROP_HIRANGE

```
const int ParseUtil::OPC_PROP_HIRANGE = 104
```

absolute maximum value -> DRVH

Definition at line 29 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.37 OPC_PROP_HTML

```
const int ParseUtil::OPC_PROP_HTML = 206
```

html file

Definition at line 40 of file ParseUtilConst.h.

6.14.2.38 OPC_PROP_INOUT

```
const int ParseUtil::OPC_PROP_INOUT = 8601
```

input or output

Definition at line 61 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY(), and EpicsTpy::epics_db_processing::operator()().

6.14.2.39 OPC_PROP_INPUT

```
const char* const ParseUtil::OPC_PROP_INPUT = "input"
```

input

Definition at line 62 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY().

6.14.2.40 OPC_PROP_LOEU

```
const int ParseUtil::OPC_PROP_LOEU = 103
```

low expectation value -> LOPR

Definition at line 28 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.41 OPC_PROP_LORANGE

```
const int ParseUtil::OPC_PROP_LORANGE = 105
```

absolute minimum value -> DRVL

Definition at line 30 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.42 OPC_PROP_OPEN

```
const int ParseUtil::OPC_PROP_OPEN = 107
```

label for open state -> ZNAM

Definition at line 32 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

6.14.2.43 OPC_PROP_OUTPUT

```
const char* const ParseUtil::OPC_PROP_OUTPUT = "output"
```

output

Definition at line 63 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY().

6.14.2.44 OPC_PROP_PINI

```
const int ParseUtil::OPC_PROP_PINI = 8603
```

initialization

Definition at line 65 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.45 OPC_PROP_PLNAME

```
const int ParseUtil::OPC_PROP_PLNAME = 8611
```

tc name including ads routing info and port

Definition at line 68 of file ParseUtilConst.h.

Referenced by ParseUtil::process_arg::get_full(), EpicsTpy::epics_db_processing::operator()(), ParseTpy::tpy_file::parse_finish(), and InfoPlc::InfoPLC::process_info().

6.14.2.46 OPC_PROP_PREC

```
const int ParseUtil::OPC_PROP_PREC = 8500
```

precision

Definition at line 57 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.47 OPC_PROP_QUALITY

const int ParseUtil::OPC_PROP_QUALITY = 3

data quality flag

Definition at line 20 of file ParseUtilConst.h.

6.14.2.48 OPC_PROP_RECTYPE

const int ParseUtil::OPC_PROP_RECTYPE = 8600

record type

Definition at line 60 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.49 OPC_PROP_RIGHTS

const int ParseUtil::OPC_PROP_RIGHTS = 5

access right: 1 read, 2 write, 3 read/write

Definition at line 22 of file ParseUtilConst.h.

Referenced by ParseUtil::opc_list::is_READONLY().

6.14.2.50 OPC_PROP_SCANRATE

const int ParseUtil::OPC_PROP_SCANRATE = 6

scan rate

Definition at line 23 of file ParseUtilConst.h.

6.14.2.51 OPC_PROP_SERVER

const int ParseUtil::OPC_PROP_SERVER = 8610

server name

Definition at line 67 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.52 OPC_PROP_SND

```
const int ParseUtil::OPC_PROP_SND = 205  
sound file
```

Definition at line 39 of file ParseUtilConst.h.

6.14.2.53 OPC_PROP_TIME

```
const int ParseUtil::OPC_PROP_TIME = 4  
timestamp
```

Definition at line 21 of file ParseUtilConst.h.

6.14.2.54 OPC_PROP_TIMEZONE

```
const int ParseUtil::OPC_PROP_TIMEZONE = 108  
time zone
```

Definition at line 33 of file ParseUtilConst.h.

6.14.2.55 OPC_PROP_TSE

```
const int ParseUtil::OPC_PROP_TSE = 8602  
time stamp
```

Definition at line 64 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.14.2.56 OPC_PROP_UNIT

```
const int ParseUtil::OPC_PROP_UNIT = 100  
unit string -> EGU
```

Definition at line 25 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

6.14.2.57 OPC_PROP_VALUE

```
const int ParseUtil::OPC_PROP_VALUE = 2  
value
```

Definition at line 19 of file ParseUtilConst.h.

6.14.2.58 OPC_PROP_ZRST

```
const int ParseUtil::OPC_PROP_ZRST = 8510  
zero string ...
```

Definition at line 58 of file ParseUtilConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

6.15 utility functions and classes

Classes

- struct `std::case_char_traits`
case insensitive traits.
- struct `std::case_wchar_traits`
case insensitive unicode traits.

Typedefs

- typedef `std::basic_string< char, case_char_traits > std::stringcase`
case insensitive string.
- typedef `std::basic_string< wchar_t, case_wchar_traits > std::wstringcase`
case insensitive string.

Functions

- int `std::strncasecmp` (const char *s1, const char *s2, size_t n)
case insensitive compare with maximum length
- int `std::wcscasewcmp` (const wchar_t *s1, const wchar_t *s2, size_t n)
case insensitive unicode compare with maximum length
- void `std::trim_space` (`std::stringcase &s`)
- void `std::trim_space` (`std::wstringcase &s`)
- template<class Container , class String , class Predicate >
void `std::split_string` (Container &output, const String &input, const Predicate &pred, bool trimEmpty=true)
Splits a strings.

6.15.1 Detailed Description

6.15.2 Typedef Documentation

6.15.2.1 `stringcase`

```
typedef std::basic_string<char, case_char_traits> std::stringcase
```

case insensitive string.

This string class is not case sensitive.

Definition at line 104 of file stringcase.h.

6.15.2.2 wstringcase

```
typedef std::basic_string<wchar_t, case_wchar_traits> std::wstringcase
```

case insensitive string.

This string class is not case sensitive.

Definition at line 109 of file stringcase.h.

6.15.3 Function Documentation

6.15.3.1 split_string()

```
template<class Container , class String , class Predicate >
void std::split_string (
    Container & output,
    const String & input,
    const Predicate & pred,
    bool trimEmpty = true )
```

Splits a strings.

Splits a string into its tokens and adds them to a container. The delimiter can be easily specified with a lambda expression. Example: stringcase arg ("This is a test!"); vector<stringcase> list; split_string (list, arg, [] (char c)->bool { return isspace (c) != 0; }, true);

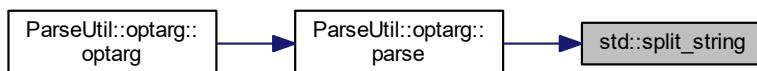
Parameters

<i>output</i>	Output container
<i>input</i>	Input string
<i>pred</i>	Function which returns true when character is a separator
<i>trimEmpty</i>	Trims empty strings when true

Definition at line 135 of file stringcase.h.

Referenced by ParseUtil::optarg::parse().

Here is the caller graph for this function:



6.15.3.2 strncasecmp()

```
int std::strncasecmp (
    const char * s1,
    const char * s2,
    size_t n ) [inline]
```

case insensitive compare with maximum length

Case insensitive compare.

Parameters

<i>s1</i>	first string
<i>s2</i>	second string
<i>n</i>	Number of characters

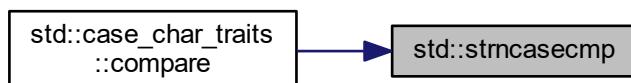
Returns

<0 smaller, 0 equal, >0 greater

Definition at line 23 of file stringcase.h.

Referenced by std::case_char_traits::compare().

Here is the caller graph for this function:



6.15.3.3 trim_space() [1/2]

```
void std::trim_space (
    std::stringcase & s )
```

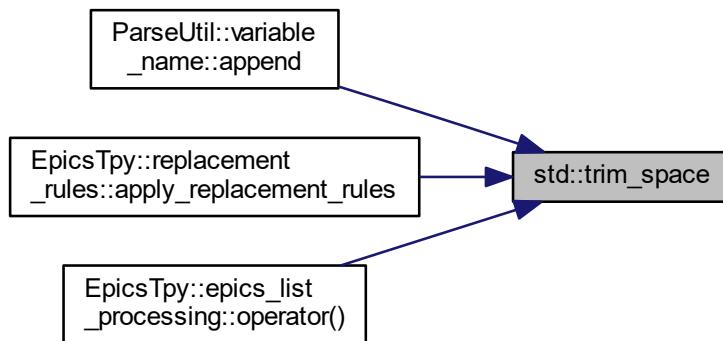
trim space on both ends.

Parameters

<i>s</i>	string to trim
----------	----------------

Referenced by ParseUtil::variable_name::append(), EpicsTpy::replacement_rules::apply_replacement_rules(), and EpicsTpy::epics_list_processing::operator()().

Here is the caller graph for this function:



6.15.3.4 trim_space() [2/2]

```
void std::trim_space (
    std::wstringcase & s )
```

trim space on both ends.

Parameters

<code>s</code>	string to trim
----------------	----------------

6.15.3.5 wcsncasewcmp()

```
int std::wcsncasewcmp (
    const wchar_t * s1,
    const wchar_t * s2,
    size_t n ) [inline]
```

case insensitive unicode compare with maximum length

Case insensitive compare for unicode string.

Parameters

<code>s1</code>	first string
<code>s2</code>	second string
<code>n</code>	Number of characters

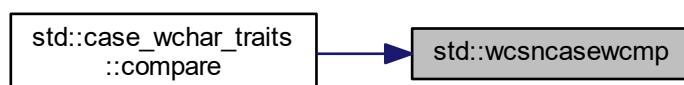
Returns

<0 smaller, 0 equal, >0 greater

Definition at line 34 of file stringcase.h.

Referenced by std::case_wchar_traits::compare().

Here is the caller graph for this function:



6.16 Utility functions and classes

Classes

- struct `std::std::hash< std::stringcase >`
hash for case insensitive string.
- struct `std::std::hash< std::wstringcase >`
hash for case insensitive unicode string.

6.16.1 Detailed Description

6.17 Classes for describing TC symbol

Classes

- struct [TcComms::DataPar](#)
Memory location struct.
- class [TcComms::TCatInterface](#)
TCat interface class.

6.17.1 Detailed Description

6.18 Classes for managing groups of TC symbols

Classes

- class [TcComms::tcProcWrite](#)
TwinCAT process write requests.
- class [TcComms::TcPLC](#)
TwinCAT PLC.
- class [TcComms::AmsRouterNotification](#)
AMS Router Notification.

6.18.1 Detailed Description

6.19 Utility functions and classes

Classes

- class `EpicsTpy::replacement_rules`
Replacement rules.
- class `EpicsTpy::epics_conversion`
Epics conversion.
- class `EpicsTpy::split_io_support`
Split IO support.
- class `EpicsTpy::multi_io_support`
Multiple IO support.

TypeDefs

- `typedef std::map< std::stringcase, std::stringcase > EpicsTpy::replacement_table`

Enumerations

- enum `EpicsTpy::tc_epics_conv` { `EpicsTpy::no_conversion`, `EpicsTpy::no_dot`, `EpicsTpy::ligo_std`, `EpicsTpy::ligo_vac` }
Conversion rules for TC/EPICS.
- enum `EpicsTpy::case_type` { `EpicsTpy::preserve_case`, `EpicsTpy::upper_case`, `EpicsTpy::lower_case` }
Case conversion rule enum.
- enum `EpicsTpy::io_filestat` { `EpicsTpy::io_filestat::closed`, `read`, `write` }
enum for file io

6.19.1 Detailed Description

6.19.2 Typedef Documentation

6.19.2.1 replacement_table

```
typedef std::map<std::stringcase, std::stringcase> EpicsTpy::replacement_table
```

Table of replacement rules

Definition at line 23 of file TpyToEpics.h.

6.19.3 Enumeration Type Documentation

6.19.3.1 case_type

```
enum EpicsTpy::case_type
```

Case conversion rule enum.

This enum describes the case conversion rule

Enumerator

preserve_case	Preserve the case.
upper_case	Convert to upper case.
lower_case	Convert to lower case.

Definition at line 88 of file TpyToEpics.h.

6.19.3.2 io_filestat

```
enum EpicsTpy::io_filestat [strong]
```

enum for file io

Enumerator

closed	file is closed
--------	----------------

Definition at line 342 of file TpyToEpics.h.

6.19.3.3 tc_epics_conv

```
enum EpicsTpy::tc_epics_conv
```

Conversion rules for TC/EPICS.

This enum describes the TwinCAT/opc to EPICS conversion rule

Enumerator

no_conversion	No conversion.
no_dot	Convert '.' to '_'.
ligo_std	LIGO standard conversion: Eliminate leading '.', Replace second '.' with ':', Replace third '.' with '-', Replace all other '.' with '-'.
ligo_vac	LIGO standard conversion for vacuum channels: Eliminate leading '.', Replace second '.' with '-', Replace third '.' with '-'.

Definition at line 66 of file TpyToEpics.h.

6.20 Classes for converting a parsed tpy

Classes

- class `EpicsTpy::epics_list_processing`
List processing.
- struct `EpicsTpy::macro_info`
Macro information.
- struct `EpicsTpy::macro_record`
Macro record.
- class `EpicsTpy::epics_macrofiles_processing`
Macro file processing.
- class `EpicsTpy::epics_db_processing`
Epics database record processing

Typedefs

- typedef `std::vector< macro_info >` `EpicsTpy::macro_list`
- typedef `std::stack< macro_record >` `EpicsTpy::macro_stack`
- typedef `std::unordered_set< std::stringcase >` `EpicsTpy::filename_set`

Enumerations

- enum `EpicsTpy::listing_type` { `EpicsTpy::listing_standard`, `EpicsTpy::listing_autoburt`, `EpicsTpy::listing_daqini` }
Listing type enum.
- enum `EpicsTpy::macrofile_type` { `EpicsTpy::macrofile_type::all`, `EpicsTpy::macrofile_type::fields`, `EpicsTpy::macrofile_type::error` }
Macrofile type enum.
- enum `EpicsTpy::device_support_type` { `EpicsTpy::device_support_opc_name`, `EpicsTpy::device_support_tc_name` }
Device support enum.

6.20.1 Detailed Description

into an EPICS database

6.20.2 Typedef Documentation

6.20.2.1 filename_set

```
typedef std::unordered_set<std::stringcase> EpicsTpy::filename_set
```

A set of filenames

Definition at line 607 of file TpyToEpics.h.

6.20.2.2 macro_list

```
typedef std::vector<macro_info> EpicsTpy::macro_list
```

A list of fields

Definition at line 580 of file TpyToEpics.h.

6.20.2.3 macro_stack

```
typedef std::stack<macro_record> EpicsTpy::macro_stack
```

A stack of records/structs

Definition at line 603 of file TpyToEpics.h.

6.20.3 Enumeration Type Documentation

6.20.3.1 device_support_type

```
enum EpicsTpy::device_support_type
```

Device support enum.

This enum describes the type of listing to produce

Enumerator

device_support_opc_name	Use opc names in the INPUT/OUTPUT epics fields.
device_support_tc_name	Use TwinCAT names in the INPUT/OUTPUT epics fields.

Definition at line 732 of file TpyToEpics.h.

6.20.3.2 listing_type

```
enum EpicsTpy::listing_type
```

Listing type enum.

This enum describes the type of listing to produce

Enumerator

listing_standard	Standard listing using TwinCAT/OPC names.
listing_autoburt	Autoburt listing.
listing_daqini	LIGO DAQ ini listing.

Definition at line 463 of file TpyToEpics.h.

6.20.3.3 macrofile_type

```
enum EpicsTpy::macrofile_type [strong]
```

This enum describes the type of macros to produce

Enumerator

all	Include all fields and error messages.
fields	Include all fields.
errors	Include error messages.

Definition at line 552 of file TpyToEpics.h.

6.21 EPICS maximum length constants

Variables

- const int `EpicsTpy::MAX_EPICS_CHANNEL` = 54
- const int `EpicsTpy::MAX_EPICS_DESC` = 40
- const int `EpicsTpy::MAX_EPICS_UNIT` = 15

6.21.1 Detailed Description

6.21.2 Variable Documentation

6.21.2.1 MAX_EPICS_CHANNEL

```
const int EpicsTpy::MAX_EPICS_CHANNEL = 54
```

maximum EPICS channel name

Definition at line 18 of file TpyToEpicsConst.h.

Referenced by `EpicsTpy::epics_db_processing::operator()()`.

6.21.2.2 MAX_EPICS_DESC

```
const int EpicsTpy::MAX_EPICS_DESC = 40
```

maximum EPICS channel name

Definition at line 19 of file TpyToEpicsConst.h.

Referenced by `EpicsTpy::epics_db_processing::operator()()`.

6.21.2.3 MAX_EPICS_UNIT

```
const int EpicsTpy::MAX_EPICS_UNIT = 15
```

maximum EPICS channel name

Definition at line 20 of file TpyToEpicsConst.h.

6.22 EPICS record field names

Variables

- const char *const EpicsTpy::EPICS_DB_EGU = "EGU"
- const char *const EpicsTpy::EPICS_DB_DESC = "DESC"
- const char *const EpicsTpy::EPICS_DB_HOPR = "HOPR"
- const char *const EpicsTpy::EPICS_DB_LOPR = "LOPR"
- const char *const EpicsTpy::EPICS_DB_DRVH = "DRVH"
- const char *const EpicsTpy::EPICS_DB_DRVL = "DRVL"
- const char *const EpicsTpy::EPICS_DB_ONAM = "ONAM"
- const char *const EpicsTpy::EPICS_DB_ZNAM = "ZNAM"
- const char *const EpicsTpy::EPICS_DB_PREC = "PREC"
- const char *const EpicsTpy::EPICS_DB_ZRST [16]
- const char *const EpicsTpy::EPICS_DB_ZRVL [16]
- const char *const EpicsTpy::EPICS_DB_SCAN = "SCAN"
- const char *const EpicsTpy::EPICS_DB_INP = "INP"
- const char *const EpicsTpy::EPICS_DB_OUT = "OUT"
- const char *const EpicsTpy::EPICS_DB_TSE = "TSE"
- const char *const EpicsTpy::EPICS_DB_PINI = "PINI"
- const char *const EpicsTpy::EPICS_DB_DTYP = "DTYP"
- const char *const EpicsTpy::EPICS_DB_OSV = "OSV"
- const char *const EpicsTpy::EPICS_DB_ZSV = "ZSV"
- const char *const EpicsTpy::EPICS_DB_COSV = "COSV"
- const char *const EpicsTpy::EPICS_DB_HIHI = "HIHI"
- const char *const EpicsTpy::EPICS_DB_HIGH = "HIGH"
- const char *const EpicsTpy::EPICS_DB_LOW = "LOW"
- const char *const EpicsTpy::EPICS_DB_LOLO = "LOLO"
- const char *const EpicsTpy::EPICS_DB_HYST = "HYST"
- const char *const EpicsTpy::EPICS_DB_HHSV = "HHSV"
- const char *const EpicsTpy::EPICS_DB_HSV = "HSV"
- const char *const EpicsTpy::EPICS_DB_LSV = "LSV"
- const char *const EpicsTpy::EPICS_DB_LLSV = "LLSV"
- const char *const EpicsTpy::EPICS_DB_NOALARM = "NO_ALARM"
- const char *const EpicsTpy::EPICS_DB_MINOR = "MINOR"
- const char *const EpicsTpy::EPICS_DB_MAJOR = "MAJOR"
- const char *const EpicsTpy::EPICS_DB_UNSV = "UNSV"
- const char *const EpicsTpy::EPICS_DB_ZRSV [16]

6.22.1 Detailed Description

6.22.2 Variable Documentation

6.22.2.1 EPICS_DB_COSV

```
const char* const EpicsTpy::EPICS_DB_COSV = "COSV"
```

change severity

Definition at line 51 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.2 EPICS_DB_DESC

```
const char* const EpicsTpy::EPICS_DB_DESC = "DESC"
```

description string

Definition at line 28 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.3 EPICS_DB_DRVH

```
const char* const EpicsTpy::EPICS_DB_DRVH = "DRVH"
```

drive high string

Definition at line 31 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.4 EPICS_DB_DRVL

```
const char* const EpicsTpy::EPICS_DB_DRVL = "DRVL"
```

drive low string

Definition at line 32 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.5 EPICS_DB_DTYP

```
const char* const EpicsTpy::EPICS_DB_DTYP = "DTYP"
```

data type

Definition at line 47 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.6 EPICS_DB_EGU

```
const char* const EpicsTpy::EPICS_DB_EGU = "EGU"
```

unit string

Definition at line 27 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.7 EPICS_DB_HHSV

```
const char* const EpicsTpy::EPICS_DB_HHSV = "HHSV"
```

high severity high limit

Definition at line 58 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.8 EPICS_DB_HIGH

```
const char* const EpicsTpy::EPICS_DB_HIGH = "HIGH"
```

low severity high limit

Definition at line 54 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.9 EPICS_DB_HIHI

```
const char* const EpicsTpy::EPICS_DB_HIHI = "HIHI"
```

high severity high limit

Definition at line 53 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.10 EPICS_DB_HOPR

```
const char* const EpicsTpy::EPICS_DB_HOPR = "HOPR"
```

high ops value string

Definition at line 29 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.11 EPICS_DB_HSV

```
const char* const EpicsTpy::EPICS_DB_HSV = "HSV"
```

low severity high limit

Definition at line 59 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.12 EPICS_DB_HYST

```
const char* const EpicsTpy::EPICS_DB_HYST = "HYST"
```

deadband

Definition at line 57 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.13 EPICS_DB_INP

```
const char* const EpicsTpy::EPICS_DB_INP = "INP"
```

input link

Definition at line 43 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.14 EPICS_DB_LLSV

```
const char* const EpicsTpy::EPICS_DB_LLSV = "LLSV"
```

high severity low limit

Definition at line 61 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.15 EPICS_DB_LOLO

```
const char* const EpicsTpy::EPICS_DB_LOLO = "LOLO"
```

high severity low limit

Definition at line 56 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.16 EPICS_DB_LOPR

```
const char* const EpicsTpy::EPICS_DB_LOPR = "LOPR"
```

low ops value string

Definition at line 30 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.17 EPICS_DB_LOW

```
const char* const EpicsTpy::EPICS_DB_LOW = "LOW"
```

low severity low limit

Definition at line 55 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.18 EPICS_DB_LSV

```
const char* const EpicsTpy::EPICS_DB_LSV = "LSV"
```

low severity low limit

Definition at line 60 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.19 EPICS_DB_MAJOR

```
const char* const EpicsTpy::EPICS_DB_MAJOR = "MAJOR"
```

major alarm

Definition at line 65 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()(), and EpicsTpy::epics_db_processing::process_field←_alarm().

6.22.2.20 EPICS_DB_MINOR

```
const char* const EpicsTpy::EPICS_DB_MINOR = "MINOR"
```

minor alarm

Definition at line 64 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()(), and EpicsTpy::epics_db_processing::process_field←_alarm().

6.22.2.21 EPICS_DB_NOALARM

```
const char* const EpicsTpy::EPICS_DB_NOALARM = "NO_ALARM"
```

no alarm

Definition at line 63 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::process_field_alarm().

6.22.2.22 EPICS_DB_ONAM

```
const char* const EpicsTpy::EPICS_DB_ONAM = "ONAM"
```

one name string

Definition at line 33 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.23 EPICS_DB_OSV

```
const char* const EpicsTpy::EPICS_DB_OSV = "OSV"
```

one severity

Definition at line 49 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.24 EPICS_DB_OUT

```
const char* const EpicsTpy::EPICS_DB_OUT = "OUT"
```

output link

Definition at line 44 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.25 EPICS_DB_PINI

```
const char* const EpicsTpy::EPICS_DB_PINI = "PINI"
```

initialization

Definition at line 46 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.26 EPICS_DB_PREC

```
const char* const EpicsTpy::EPICS_DB_PREC = "PREC"
```

precision

Definition at line 36 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.27 EPICS_DB_SCAN

```
const char* const EpicsTpy::EPICS_DB_SCAN = "SCAN"
```

Scan

Definition at line 42 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.28 EPICS_DB_TSE

```
const char* const EpicsTpy::EPICS_DB_TSE = "TSE"
```

time stamp

Definition at line 45 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.29 EPICS_DB_UNSV

```
const char* const EpicsTpy::EPICS_DB_UNSV = "UNSV"
```

unknown severity

Definition at line 67 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.30 EPICS_DB_ZNAM

```
const char* const EpicsTpy::EPICS_DB_ZNAM = "ZNAM"
```

zero name string

Definition at line 34 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.31 EPICS_DB_ZRST

```
const char* const EpicsTpy::EPICS_DB_ZRST[16]
```

Initial value:

```
= {"ZRST", "ONST", "TWST", "THST", "FRST", "FVST", "SXST", "SVST",
     "EIST", "NIST", "TEST", "ELST", "TVST", "TTST", "FTST", "FFST"}
```

enum string

Definition at line 37 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.32 EPICS_DB_ZRSV

```
const char* const EpicsTpy::EPICS_DB_ZRSV[16]
```

Initial value:

```
= {"ZRSV", "ONSV", "TWSV", "THSV", "FRSV", "FVSV", "SXSV", "SVSV",
     "EISV", "NISV", "TESV", "ELSV", "TVSV", "TTSV", "FTSV", "FFSV"}
```

enum alarm severity

Definition at line 68 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.33 EPICS_DB_ZRVL

```
const char* const EpicsTpy::EPICS_DB_ZRVL[16]
```

Initial value:

```
= {"ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL",
     "EIVL", "NIVL", "TEVL", "ELVL", "TVVL", "TTVL", "FTVL", "FFVL"}
```

enum val

Definition at line 39 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.22.2.34 EPICS_DB_ZSV

```
const char* const EpicsTpy::EPICS_DB_ZSV = "ZSV"
```

zero severity

Definition at line 50 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_db_processing::operator()().

6.23 Lists of EPICS record field names

Variables

- const char *const EpicsTpy::EPICS_DB_FORBIDDEN []
- const char *const EpicsTpy::EPICS_DB_ALLOWED []
- const char *const EpicsTpy::EPICS_DB_NUMVAL []

6.23.1 Detailed Description

used to describe allowed, forbidden and numeric fields

6.23.2 Variable Documentation

6.23.2.1 EPICS_DB_ALLOWED

```
const char* const EpicsTpy::EPICS_DB_ALLOWED[]
```

Initial value:

```
= {"LINR", "EGUF", "EGUL", "AOFF", "ASLO", "ESLO", "SMOO", "HIHI", "LOLO", "HIGH", "LOW", "HHSV", "LLSV",
    "HSV", "LSV", "HYST", "ZSV", "OSV", "COSV", "IVOA", "IVOV", "UNSV", "ZRSV",
    "ONSV", "TWSV", "THSV",
    "FRSV", "FVSV", "SXSV", "SVSV", "EISV", "NISV", "TESV", "ELSV", "TVSV", "TTSV",
    "FTSV", "FFSV", NULL}
```

Names of allowed EPICS record fields

Definition at line 85 of file TpyToEpicsConst.h.

6.23.2.2 EPICS_DB_FORBIDDEN

```
const char* const EpicsTpy::EPICS_DB_FORBIDDEN[]
```

Initial value:

```
= {"", EPICS_DB_SCAN, "DOL", EPICS_DB_INP, EPICS_DB_OUT, "VAL", "RVAL", "INIT",
    "ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL",
    "EIVL", "NIVL", "TEVL", "ELVL", "TVVL", "TTVL", "FTVL", "FFVL", NULL}
```

Names of forbidden EPICS record fields

Definition at line 79 of file TpyToEpicsConst.h.

6.23.2.3 EPICS_DB_NUMVAL

```
const char* const EpicsTpy::EPICS_DB_NUMVAL[]
```

Initial value:

```
= {EPICS_DB_TSE, EPICS_DB_HOPR, EPICS_DB_LOPR, EPICS_DB_DRVH, EPICS_DB_DRVL, "EGUF", "EGUL",
    "AOFF", "ASLO", "ESLO", "SMOO", "PREC", "HIHI", "LOLO", "HIGH", "LOW", "HYST",
    "ZRVL", "ONVL", "TWVL", "THVL", "FRVL", "FVVL", "SXVL", "SVVL", "EIVL", "NIVL",
    "TEVL", "ELVL",
    "TVVL", "TTVL", "FTVL", "FFVL", NULL}
```

Names of EPICS record fields which are numeric

Definition at line 91 of file TpyToEpicsConst.h.

6.24 LIGO related constants

Variables

- const char *const EpicsTpy::LIGODAQ_DATATYPE_NAME = "datatype"
- const int EpicsTpy::LIGODAQ_DATATYPE_FLOAT = 4
- const int EpicsTpy::LIGODAQ_DATATYPE_INT32 = 2
- const int EpicsTpy::LIGODAQ_DATATYPE_DEFAULT = LIGODAQ_DATATYPE_FLOAT
- const char *const EpicsTpy::LIGODAQ_UNIT_NAME = "units"
- const char *const EpicsTpy::LIGODAQ_UNIT_NONE = "none"
- const char *const EpicsTpy::LIGODAQ_UNIT_DEFAULT = LIGODAQ_UNIT_NONE
- const char *const EpicsTpy::LIGODAQ_INI_HEADER

6.24.1 Detailed Description

6.24.2 Variable Documentation

6.24.2.1 LIGODAQ_DATATYPE_DEFAULT

```
const int EpicsTpy::LIGODAQ_DATATYPE_DEFAULT = LIGODAQ_DATATYPE_FLOAT
```

DAQ datatype is float

Definition at line 105 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.24.2.2 LIGODAQ_DATATYPE_FLOAT

```
const int EpicsTpy::LIGODAQ_DATATYPE_FLOAT = 4
```

DAQ datatype is float

Definition at line 103 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.24.2.3 LIGODAQ_DATATYPE_INT32

```
const int EpicsTpy::LIGODAQ_DATATYPE_INT32 = 2
```

DAQ datatype is int32

Definition at line 104 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.24.2.4 LIGODAQ_DATATYPE_NAME

const char* const EpicsTpy::LIGODAQ_DATATYPE_NAME = "datatype"

DAQ datatype name

Definition at line 102 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.24.2.5 LIGODAQ_INI_HEADER

const char* const EpicsTpy::LIGODAQ_INI_HEADER

Initial value:

```
=
"[default]\n"
"gain=1.00\n"
"datatype=%i\n"
"ifoid=0\n"
"slope=6.1028e-05\n"
"acquire=3\n"
"offset=0\n"
"units=%s\n"
"dcuid=4\n"
"datarate=16"
```

DAQ ini file header: substitute defaults

Definition at line 112 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.24.2.6 LIGODAQ_UNIT_DEFAULT

const char* const EpicsTpy::LIGODAQ_UNIT_DEFAULT = [LIGODAQ_UNIT_NONE](#)

default DAQ unit is none

Definition at line 109 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.24.2.7 LIGODAQ_UNIT_NAME

const char* const EpicsTpy::LIGODAQ_UNIT_NAME = "units"

DAQ unit name

Definition at line 107 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

6.24.2.8 LIGODAQ_UNIT_NONE

const char* const EpicsTpy::LIGODAQ_UNIT_NONE = "none"

DAQ unit for no unit

Definition at line 108 of file TpyToEpicsConst.h.

Referenced by EpicsTpy::epics_list_processing::operator()().

Chapter 7

Namespace Documentation

7.1 DevInfo Namespace Reference

Namespace for info device support.

Classes

- class `epics_info_db_processing`
EPICS/Info db processing.
- class `InfoRegisterTolocShell`
Register info commands.
- class `register_info_devsup`
Epics interface class.

TypeDefs

- `typedef std::pair< std::stringcase, std::stringcase > filename_rule_pair`
filename/rule pair

Functions

- `bool linkInfoRecord (dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`
Link Information Record.
- `const std::regex info_regex ("((info)::((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?)|(8[0-9][0-9]))|((\\d{1,9})|((\\d{1,9}):(\\d{1,9})))")`
Regex for indentifying TwinCAT records.
- `void infoLoadRecords (const iocshArgBuf *args)`
Info load records.
- `void infoSetScanRate (const iocshArgBuf *args)`
Info set scan rate.
- `void infoList (const iocshArgBuf *args)`
channel lists
- `void infoAlias (const iocshArgBuf *args)`
alias
- `void infoPrefix (const iocshArgBuf *args)`
tag prefix
- `void infoPrintVals (const iocshArgBuf *args)`
Info print vals.

7.1.1 Detailed Description

Namespace for info device support.

[DevInfo](#) namespace

7.1.2 Typedef Documentation

7.1.2.1 filename_rule_pair

```
typedef std::pair<std::stringcase, std::stringcase> DevInfo::filename_rule_pair
```

filename/rule pair

Class for storing a flanem and a rule

Definition at line 61 of file `drvInfo.cpp`.

7.1.3 Function Documentation

7.1.3.1 linkInfoRecord()

```
bool DevInfo::linkInfoRecord (
    dbCommon * pEpicsRecord,
    plc::BaseRecordPtr & pRecord )
```

Link Information Record.

Initialization function that matches an EPICS record with an internal Info record entry

Parameters

<i>pEpicsRecord</i>	Pointer to EPICS record
<i>pRecord</i>	Pointer to a base record

Returns

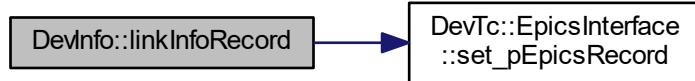
true if successful

Definition at line 33 of file `devInfo.cpp`.

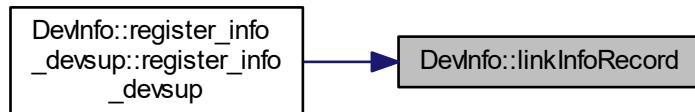
References `DevTc::EpicsInterface::set_pEpicsRecord()`.

Referenced by `DevInfo::register_info_devsup::register_info_devsup()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.2 DevTc Namespace Reference

Namespace for info device support.

Classes

- struct [devTcDefIn](#)
Device support input record.
- struct [devTcDefLo](#)
Device support record.
- struct [devTcDefOut](#)
device support output record.
- struct [devTcDefWaveformIn](#)
device support waveform record.
- struct [epics_record_traits](#)
Epics record traits.
- class [epics_tc_db_processing](#)
EPICS/TCat db processing.
- class [EpicsInterface](#)
Epics interface class.
- class [register_devsup](#)
Device support registration.
- class [tcRegisterTolocShell](#)
Register TC commands.

Typedefs

- `typedef std::tuple< std::stringcase, std::stringcase, epics_list_processing *, bool > filename_rule_list_tuple`
Tuple for filnemae, rule and list processing.
- `typedef std::vector< filename_rule_list_tuple > tc_listing_def`
List of tuples for filnemae, rule and list processing.
- `typedef std::tuple< std::stringcase, std::stringcase, epics_macrofiles_processing *, const char * > dirname_arg_macro_tuple`
Tuple for directory name, argument and macro list processing.
- `typedef std::vector< dirname_arg_macro_tuple > tc_macro_def`
List of tuples for directory name, argument and macro list processing.

Enumerations

- `enum epics_record_enum {
 aaival = 0, aaoval, aival, aoval,
 bival, boval, eventval, histogramval,
 longinval, longoutval, mbbival, mbboval,
 mbbiDirectval, mbboDirectval, stringinval, stringoutval,
 waveformval, airval, aorval, birval,
 borval, mbbiDirectrval, mbboDirectrval, mbbirval,
 mbborval, epics_record_enumEnd, invalidval = -1 }`

Epics record type enum.

Functions

- `bool linkRecord (std::stringcase name, dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`
Link Record.
- `bool linkTcRecord (dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`
Link TwinCat Record.
- `void outRecordCallback (callbackPvt *pcallback)`
Callback for output record.
- `const std::regex tc_regex ("((tc)::((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?)|(8[0-9][0-9]))|((\\d{1,9})|((\\d{1,9}):(\\d{1,9})))")`
Regex for indentifying TwinCAT records.
- `void tcLoadRecords (const iocshArgBuf *args)`
TCat load records.
- `void tcSetScanRate (const iocshArgBuf *args)`
TCat set scan rate.
- `void tcList (const iocshArgBuf *args)`
channel lists
- `void tcMacro (const iocshArgBuf *args)`
macro files
- `void tcAlias (const iocshArgBuf *args)`
alias
- `void tcPrintVals (const iocshArgBuf *args)`
TCat print vals.

7.2.1 Detailed Description

Namespace for info device support.

Namespace for TCat device support.

[DevTc](#) name space

[DevTc](#) Name space

[DevTc](#) namespace

7.2.2 Function Documentation

7.2.2.1 linkRecord()

```
bool DevTc::linkRecord (
    std::stringcase name,
    dbCommon * pEpicsRecord,
    BaseRecordPtr & pRecord )
```

Link Record.

linkRecord

Initialization function that matches an EPICS record with an internal record entry

Parameters

<i>name</i>	Name of record (INP/OUT field)
<i>pEpicsRecord</i>	Pointer to EPICS record
<i>pRecord</i>	Pointer to a base record

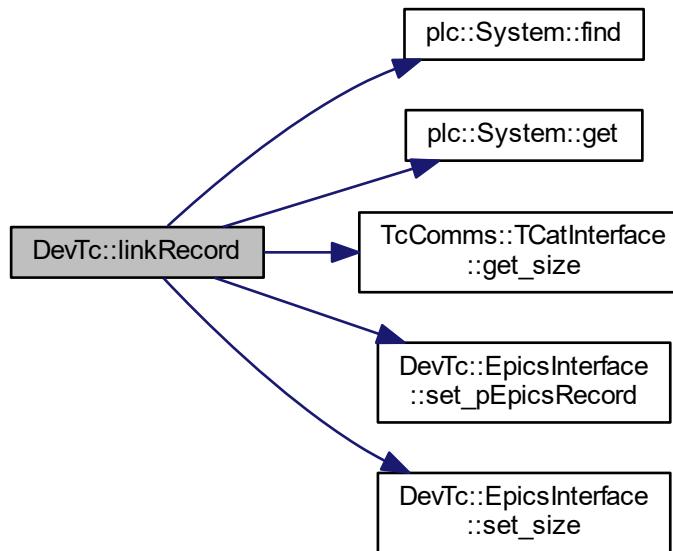
Returns

true if successful

Definition at line 34 of file devTc.cpp.

References `plc::System::find()`, `plc::System::get()`, `TcComms::TCatInterface::get_size()`, `DevTc::EpicsInterface::set_pEpicsRecord()`, and `DevTc::EpicsInterface::set_size()`.

Here is the call graph for this function:



7.2.2.2 linkTcRecord()

```
bool DevTc::linkTcRecord (
    dbCommon * pEpicsRecord,
    plc::BaseRecordPtr & pRecord )
```

Link TwinCat Record.

`linkTcRecord`

Initialization function that matches an EPICS record with an internal TwinCAT record entry

Parameters

<code>pEpicsRecord</code>	Pointer to EPICS record
<code>pRecord</code>	Pointer to a base record

Returns

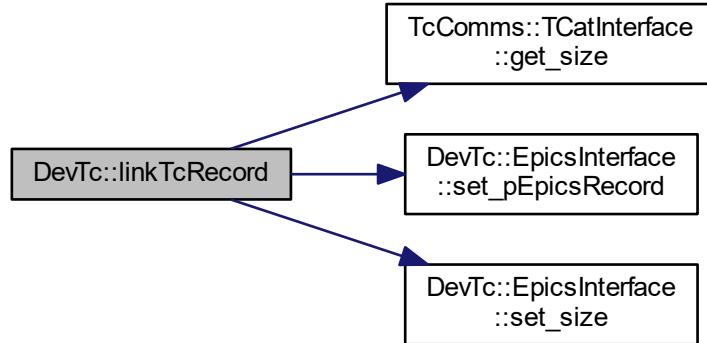
true if successful

Definition at line 86 of file devTc.cpp.

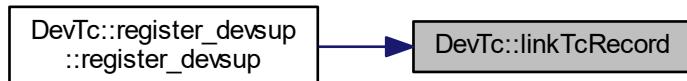
References `TcComms::TCatInterface::get_size()`, `DevTc::EpicsInterface::set_pEpicsRecord()`, and `DevTc::EpicsInterface::set_size()`.

Referenced by DevTc::register_devsup::register_devsup().

Here is the call graph for this function:



Here is the caller graph for this function:



7.2.2.3 outRecordCallback()

```
void DevTc::outRecordCallback (
    callbackPvt * pcallback ) [inline]
```

Callback for output record.

Callback function to process EPICS out recordCallback function for output records

Definition at line 30 of file devTc.h.

7.3 EpicsTpy Namespace Reference

Namespace for tpy-db conversion.

Classes

- class [epics_conversion](#)
Epics conversion.
- class [epics_db_processing](#)
pics database record processing
- class [epics_list_processing](#)
List processing.
- class [epics_macrofiles_processing](#)
Macro file processing.
- struct [macro_info](#)
Macro information.
- struct [macro_record](#)
Macro record.
- class [multi_io_support](#)
Multiple IO support.
- class [replacement_rules](#)
Replacement rules.
- class [split_io_support](#)
Split IO support.

Typedefs

- typedef std::map< std::stringcase, std::stringcase > [replacement_table](#)
- typedef std::vector< [macro_info](#) > [macro_list](#)
- typedef std::stack< [macro_record](#) > [macro_stack](#)
- typedef std::unordered_set< std::stringcase > [filename_set](#)

Enumerations

- enum [tc_epics_conv](#) { [no_conversion](#), [no_dot](#), [ligo_std](#), [ligo_vac](#) }
Conversion rules for TC/EPICS.
- enum [case_type](#) { [preserve_case](#), [upper_case](#), [lower_case](#) }
Case conversion rule enum.
- enum [io_filestat](#) { [io_filestat::closed](#), [read](#), [write](#) }
enum for file io
- enum [listing_type](#) { [listing_standard](#), [listing_autoburt](#), [listing_daqini](#) }
Listing type enum.
- enum [macrofile_type](#) { [macrofile_type::all](#), [macrofile_type::fields](#), [macrofile_type::errors](#) }
- enum [device_support_type](#) { [device_support_opc_name](#), [device_support_tc_name](#) }
Device support enum.

Variables

- const int `MAX_EPICS_CHANNEL` = 54
- const int `MAX_EPICS_DESC` = 40
- const int `MAX_EPICS_UNIT` = 15
- const char *const `EPICS_DB_EGU` = "EGU"
- const char *const `EPICS_DB_DESC` = "DESC"
- const char *const `EPICS_DB_HOPR` = "HOPR"
- const char *const `EPICS_DB_LOPR` = "LOPR"
- const char *const `EPICS_DB_DRVH` = "DRVH"
- const char *const `EPICS_DB_DRVL` = "DRVL"
- const char *const `EPICS_DB_ONAM` = "ONAM"
- const char *const `EPICS_DB_ZNAM` = "ZNAM"
- const char *const `EPICS_DB_PREC` = "PREC"
- const char *const `EPICS_DB_ZRST` [16]
- const char *const `EPICS_DB_ZRVL` [16]
- const char *const `EPICS_DB_SCAN` = "SCAN"
- const char *const `EPICS_DB_INP` = "INP"
- const char *const `EPICS_DB_OUT` = "OUT"
- const char *const `EPICS_DB_TSE` = "TSE"
- const char *const `EPICS_DB_PINI` = "PINI"
- const char *const `EPICS_DB_DTYP` = "DTYP"
- const char *const `EPICS_DB_OSV` = "OSV"
- const char *const `EPICS_DB_ZSV` = "ZSV"
- const char *const `EPICS_DB_COSV` = "COSV"
- const char *const `EPICS_DB_HIHI` = "HIHI"
- const char *const `EPICS_DB_HIGH` = "HIGH"
- const char *const `EPICS_DB_LOW` = "LOW"
- const char *const `EPICS_DB_LOLO` = "LOLO"
- const char *const `EPICS_DB_HYST` = "HYST"
- const char *const `EPICS_DB_HHSV` = "HHSV"
- const char *const `EPICS_DB_HSV` = "HSV"
- const char *const `EPICS_DB_LSV` = "LSV"
- const char *const `EPICS_DB_LLSV` = "LLSV"
- const char *const `EPICS_DB_NOALARM` = "NO_ALARM"
- const char *const `EPICS_DB_MINOR` = "MINOR"
- const char *const `EPICS_DB_MAJOR` = "MAJOR"
- const char *const `EPICS_DB_UNSV` = "UNSV"
- const char *const `EPICS_DB_ZRSV` [16]
- const char *const `EPICS_DB_FORBIDDEN` []
- const char *const `EPICS_DB_ALLOWED` []
- const char *const `EPICS_DB_NUMVAL` []
- const char *const `LIGODAQ_DATATYPE_NAME` = "datatype"
- const int `LIGODAQ_DATATYPE_FLOAT` = 4
- const int `LIGODAQ_DATATYPE_INT32` = 2
- const int `LIGODAQ_DATATYPE_DEFAULT` = `LIGODAQ_DATATYPE_FLOAT`
- const char *const `LIGODAQ_UNIT_NAME` = "units"
- const char *const `LIGODAQ_UNIT_NONE` = "none"
- const char *const `LIGODAQ_UNIT_DEFAULT` = `LIGODAQ_UNIT_NONE`
- const char *const `LIGODAQ_INI_HEADER`

7.3.1 Detailed Description

Namespace for tpy-db conversion.

[EpicsTpy](#) name space

7.4 InfoPlc Namespace Reference

Namespace for Info communication.

Classes

- class [BaseInfoItem](#)
Base info item.
- class [HistogramInfoItem](#)
Histogram statistic.
- class [HistoryInfoItem](#)
History tracker.
- class [InfoInterface](#)
Info interface.
- class [InfoPLC](#)
Info plc.
- class [SimpleInfoItem](#)
Simple info tracker.
- struct [stat_value](#)
Statistic value.
- class [TimeInfoItem](#)
Timing tracker.

Typedefs

- typedef std::vector< [stat_value](#) > [stat_list](#)
Statistic list.
- typedef std::shared_ptr< [BaseInfoItem](#) > [BaseInfoItemPtr](#)
Smart pointer to Base info items.
- typedef std::vector< [BaseInfoItemPtr](#) > [BaseInfoList](#)
List of Base info items.

Variables

- const int [default_scanrate](#) = 100
default PLC TwinCAT scan rate (100ms)
- const int [minimum_scanrate](#) = 5
minimum PLC TwinCAT scan rate (5ms)
- const int [maximum_scanrate](#) = 10000
maximum PLC TwinCAT scan rate (10s)

7.4.1 Detailed Description

Namespace for Info communication.

General plan:

- [InfoPLC](#) is created along with System
- InfoItems are created and updated at appropriate points in the other parts of the code
- an InfoItem creates several InfoInterfaces with the proper epics_params_structs, and a Record for each, adds these to [InfoPLC](#)
- infoLoad() is called from startup file after [tcLoadRecords\(\)](#)
- infoLoad() generates the dB file from the contents of [InfoPLC](#)
- See [infoPLC.cpp](#), the [SimpleInfoItem](#) stuff to see what "InfoItem" is intended to do [InfoPlc](#) name space, which has all the classes and functions used for communicating with Info.

7.4.2 Typedef Documentation

7.4.2.1 stat_list

```
typedef std::vector<stat_value> InfoPlc::stat_list
```

Statistic list.

This is a list of statistic value

Definition at line 56 of file infoPlc.h.

7.5 ParseTpy Namespace Reference

Namespace for parsing.

Classes

- class [ads_routing_info](#)
ADS routing information.
- class [base_record](#)
Base record definition.
- class [compiler_info](#)
Compiler information.
- class [item_record](#)
item record
- class [parserinfo_type](#)
- class [project_record](#)
Project information.
- class [symbol_record](#)
Symbol record.
- class [tpy_file](#)
Tpy file parsing.
- class [type_map](#)
Type dictionary.
- class [type_record](#)

Typedefs

- `typedef std::pair< int, int > dimension`
- `typedef std::list< dimension > dimensions`
- `typedef std::map< int, std::stringcase > enum_map`
- `typedef std::pair< int, std::stringcase > enum_pair`
- `typedef std::list< item_record > item_list`
- `typedef std::multimap< unsigned int, type_record > type_multipmap`
- `typedef std::list< symbol_record > symbol_list`

Enumerations

- `enum type_enum {
 unknown, simple, arraytype, enumtype,
 structtype, functionblock }`

Type enum.

Functions

- `bool compareNamesWoNamespace (const std::stringcase &p1, const std::stringcase &p2)`
- `bool get_decoration (const char **atts, unsigned int &decoration)`
- `bool get_pointer (const char **atts)`

Variables

- `const char *const xmlPlcProjectInfo = "PlcProjectInfo"`
PLC project info.
- `const char *const xmlProjectInfo = "ProjectInfo"`
Project info.
- `const char *const xmlRoutingInfo = "RoutingInfo"`
Routing info.
- `const char *const xmlCompilerInfo = "CompilerInfo"`
Compiler info.
- `const char *const xmlAdsInfo = "AdsInfo"`
ADS info.
- `const char *const xmlDataTypes = "DataTypes"`
Data types.
- `const char *const xmlDataType = "DataType"`
Data type.
- `const char *const xmlSymbols = "Symbols"`
Symbols.
- `const char *const xmlSymbol = "Symbol"`
Symbol.
- `const char *const xmlProperties = "Properties"`
Properties.
- `const char *const xmlProperty = "Property"`
Property.
- `const char *const xmlCompilerVersion = "CompilerVersion"`
Compiler version.
- `const char *const xmlTwinCATVersion = "TwinCATVersion"`

- TwinCAT version.*
- const char *const **xmlCpuFamily** = "CpuFamily"
CPU family.
 - const char *const **xmlNetId** = "NetId"
Net ID.
 - const char *const **xmlPort** = "Port"
Port.
 - const char *const **xmlTargetName** = "TargetName"
Target name.
 - const char *const **xmlName** = "Name"
Name.
 - const char *const **xmlType** = "Type"
Type.
 - const char *const **xmlAttrDecoration** = "Decoration"
Decoration.
 - const char *const **xmlAttrPointer** = "Pointer"
Pointer.
 - const char *const **xmlIIGroup** = "IGroup"
I Group.
 - const char *const **xmlIOffset** = "IOffset"
I Offset.
 - const char *const **xmlBitSize** = "BitSize"
Bit size.
 - const char *const **xmlBitOffs** = "BitOffs"
Bit Offset.
 - const char *const **xmlArrayInfo** = "ArrayInfo"
Array info.
 - const char *const **xmlArrayLBound** = "LBound"
Lower bound.
 - const char *const **xmlArrayElements** = "Elements"
Elements.
 - const char *const **xmlSubItem** = "SubItem"
Sub item.
 - const char *const **xmlFbInfo** = "FbInfo"
Fb info.
 - const char *const **xmlEnumInfo** = "EnumInfo"
Enum info.
 - const char *const **xmlEnumText** = "Text"
Text.
 - const char *const **xmlEnumEnum** = "Enum"
Enum.
 - const char *const **xmlEnumComment** = "Comment"
Comment.
 - const char *const **xmlValue** = "Value"
Value.
 - const char *const **xmlDesc** = "Desc"
Description.
 - const char *const **opcExport** = "opc"
OPC.
 - const char *const **opcProp** = "opc_prop"
OPC property.
 - const char *const **opcBracket** = "["
OPC bracket.

7.5.1 Detailed Description

Namespace for parsing.

Namespace for tpy parsing.

[ParseTpy](#) name space

7.5.2 Function Documentation

7.5.2.1 compareNamesWoNamespace()

```
bool ParseTpy::compareNamesWoNamespace (
    const std::stringcase & p1,
    const std::stringcase & p2 )
```

compareNamesWoNamespace

Definition at line 362 of file ParseTpy.cpp.

7.5.2.2 get_decoration()

```
bool ParseTpy::get_decoration (
    const char ** attrs,
    unsigned int & decoration )
```

XML get decoration number from attribute

Definition at line 497 of file ParseTpy.cpp.

References [xmlAttrDecoration](#).

7.5.2.3 get_pointer()

```
bool ParseTpy::get_pointer (
    const char ** attrs )
```

XML get pointer from attribute

Definition at line 511 of file ParseTpy.cpp.

References [xmlAttrPointer](#).

7.6 ParseUtil Namespace Reference

Namespace for parsing utilities.

Classes

- class [bit_location](#)
Bit location.
- class [memory_location](#)
Memory location.
- class [opc_list](#)
OPC list.
- class [optarg](#)
Optional arguments.
- class [process_arg](#)
Arguments for processing.
- class [tag_processing](#)
Tag processing selection.
- class [variable_name](#)
Variable name.

Typedefs

- `typedef std::map< int, std::stringcase > property_map`
- `typedef std::pair< int, std::stringcase > property_el`

Enumerations

- enum [opc_enum](#) { `no_change`, `publish`, `silent` }
OPC state enum.
- enum [process_type_enum](#) {
 `pt_invalid`, `pt_int`, `pt_real`, `pt_bool`,
 `pt_string`, `pt_enum`, `pt_binary` }
Process type.
- enum [process_tag_enum](#) { `process_all`, `process_atomic`, `process_structured` }
Tag preoicessing enum.

Variables

- const int [OPC_PROP_CDT](#) = 1
- const int [OPC_PROP_VALUE](#) = 2
- const int [OPC_PROP_QUALITY](#) = 3
- const int [OPC_PROP_TIME](#) = 4
- const int [OPC_PROP_RIGHTS](#) = 5
- const int [OPC_PROP_SCANRATE](#) = 6
- const int [OPC_PROP_UNIT](#) = 100
- const int [OPC_PROP_DESC](#) = 101
- const int [OPC_PROP_HIEU](#) = 102

- const int `OPC_PROP_LOEU` = 103
- const int `OPC_PROP_HIRANGE` = 104
- const int `OPC_PROP_LORANGE` = 105
- const int `OPC_PROP_CLOSE` = 106
- const int `OPC_PROP_OPEN` = 107
- const int `OPC_PROP_TIMEZONE` = 108
- const int `OPC_PROP_FGC` = 201
- const int `OPC_PROP_BGC` = 202
- const int `OPC_PROP_BLINK` = 203
- const int `OPC_PROP_BMP` = 204
- const int `OPC_PROP SND` = 205
- const int `OPC_PROP_HTML` = 206
- const int `OPC_PROP_AVI` = 207
- const int `OPC_PROP_ALMSTAT` = 300
- const int `OPC_PROP_ALMHELP` = 301
- const int `OPC_PROP_ALMAREAS` = 302
- const int `OPC_PROP_ALMPRIMARYAREA` = 303
- const int `OPC_PROP_ALMCONDITION` = 304
- const int `OPC_PROP_ALMLIMIT` = 305
- const int `OPC_PROP_ALMDB` = 306
- const int `OPC_PROP_ALMH` = 307
- const int `OPC_PROP_ALMH` = 308
- const int `OPC_PROP_ALML` = 309
- const int `OPC_PROP_ALMLL` = 310
- const int `OPC_PROP_ALMROC` = 311
- const int `OPC_PROP_ALMDEV` = 312
- const int `OPC_PROP_PREC` = 8500
- const int `OPC_PROP_ZRST` = 8510
- const int `OPC_PROP_FFST` = 8525
- const int `OPC_PROP_RECTYPE` = 8600
- const int `OPC_PROP_INOUT` = 8601
- const char *const `OPC_PROP_INPUT` = "input"
- const char *const `OPC_PROP_OUTPUT` = "output"
- const int `OPC_PROP_TSE` = 8602
- const int `OPC_PROP_PINI` = 8603
- const int `OPC_PROP_DTYP` = 8604
- const int `OPC_PROP_SERVER` = 8610
- const int `OPC_PROP_PLNAME` = 8611
- const int `OPC_PROP_ALIAS` = 8620
- const int `OPC_PROP_ALMOSV` = 8700
- const int `OPC_PROP_ALMZSV` = 8701
- const int `OPC_PROP_ALMCOSV` = 8702
- const int `OPC_PROP_ALMUNSV` = 8703
- const int `OPC_PROP_ALMZRSV` = 8710
- const int `OPC_PROP_ALMFFSV` = 8725
- const int `OPC_PROP_ALMHHSV` = 8727
- const int `OPC_PROP_ALMHHSV` = 8728
- const int `OPC_PROP_ALMLSV` = 8729
- const int `OPC_PROP_ALMLLSV` = 8730

7.6.1 Detailed Description

Namespace for parsing utilities.

`ParseUtil` name space

`ParseTpy` name space

7.7 plc Namespace Reference

Namespace for abstract plc functionality.

Classes

- class [BasePLC](#)
Base PLC.
- class [BaseRecord](#)
Class for managing a tag/channel.
- class [DataValue](#)
Data value.
- struct [DataValueTraits](#)
Data value traits.
- struct [DataValueTypeDef](#)
Collection of type definitions.
- class [Interface](#)
Abstract interface.
- struct [scanner_thread_args](#)
- class [System](#)
System.

Typedefs

- typedef std::shared_ptr< [BasePLC](#) > [BasePLCPtr](#)
Smart pointer to PLC.
- typedef std::unique_ptr< [Interface](#) > [InterfacePtr](#)
Smart pointer to interface.
- typedef std::shared_ptr< [BaseRecord](#) > [BaseRecordPtr](#)
smart pointer to record
- typedef std::unordered_map< std::stringcase, [BaseRecordPtr](#) > [BaseRecordList](#)
list of record
- typedef std::map< std::stringcase, [BasePLCPtr](#) > [BasePLCList](#)
BasePLC map.

Enumerations

- enum [data_type_enum](#) {
 dtInvalid, dtBool, dtInt8, dtUInt8,
 dtInt16, dtUInt16, dtInt32, dtUInt32,
 dtInt64, dtUInt64, dtFloat, dtDouble,
 dtString, dtWString, dtBinary }
Data type enumeration.
- enum [access_rights_enum](#) { [read_only](#), [write_only](#), [read_write](#) }
Access rights enum.

Functions

- template<>
`bool reset_and_read (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::type_wstring &dest, DataValueTypeDef::atomic_string *source)`
Reset and read.
- template<>
`bool write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, DataValueTypeDef::atomic_wstring *dest, const DataValueTypeDef::type_string &source)`
Write and test.
- VOID CALLBACK ScannerProc (LPVOID lpArg, DWORD dwTimerLowValue, DWORD dwTimerHighValue)
- DWORD WINAPI scannerThread (scanner_thread_args args)
- template<typename T, typename U>
`bool reset_and_read (DataValueTypeDef::atomic_bool &dirty, T &dest, U source)`
Reset and read.
- template<typename T>
`bool reset_and_read (DataValueTypeDef::atomic_bool &dirty, T &dest, typename DataValueTraits< T >::traits_atomic *source)`
Reset and read.
- template<typename T, typename U>
`bool write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, U dest, const T &source)`
Write and test.
- template<typename T>
`bool write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, typename DataValueTraits< T >::traits_atomic *dest, const T &source)`
Write and test.

7.7.1 Detailed Description

Namespace for abstract plc functionality.

PLC namespace

7.7.2 Typedef Documentation

7.7.2.1 BasePLCList

```
typedef std::map<std::stringcase, BasePLCPtr> plc::BasePLCList
```

BasePLC map.

This is list of `BasePLC`, ordered by their name.

Definition at line 881 of file `plcBase.h`.

7.7.2.2 BasePLCPtr

```
typedef std::shared_ptr<BasePLC> plc::BasePLCPtr
```

Smart pointer to PLC.

This is a smart pointer to a PLC

Definition at line 20 of file plcBase.h.

7.7.2.3 BaseRecordList

```
typedef std::unordered_map<std::stringcase, BaseRecordPtr> plc::BaseRecordList
```

list of record

This is a list of tag/channel records organized as a hash map

Definition at line 701 of file plcBase.h.

7.7.2.4 BaseRecordPtr

```
typedef std::shared_ptr<BaseRecord> plc::BaseRecordPtr
```

smart pointer to record

This is a smart pointer to a tag/channel record

Definition at line 696 of file plcBase.h.

7.7.2.5 InterfacePtr

```
typedef std::unique_ptr<Interface> plc::InterfacePtr
```

Smart pointer to interface.

This is a smart pointer for [Interface](#)

Definition at line 60 of file plcBase.h.

7.7.3 Enumeration Type Documentation

7.7.3.1 access_rights_enum

```
enum plc::access_rights_enum
```

Access rights enum.

Enum for access rights of a record

Enumerator

<code>read_only</code>	Read only.
<code>write_only</code>	Write only.
<code>read_write</code>	Read/write.

Definition at line 470 of file plcBase.h.

7.7.3.2 data_type_enum

```
enum plc::data\_type\_enum
```

Data type enumeration.

This is an enumerated type listing all the available data types

Enumerator

<code>dtInvalid</code>	Invalid data type.
<code>dtBool</code>	Boolean.
<code>dtInt8</code>	1-byte integer
<code>dtUInt8</code>	1-byte unsigned integer
<code>dtInt16</code>	2-byte integer
<code>dtUInt16</code>	2-byte unsigned integer
<code>dtInt32</code>	4-byte integer
<code>dtUInt32</code>	4-byte unsigned integer
<code>dtInt64</code>	8-byte integer
<code>dtUInt64</code>	8-byte unsigned integer
<code>dtFloat</code>	4-byte single precision floating point
<code>dtDouble</code>	8-byte double precision floating point
<code>dtString</code>	string class
<code>dtWString</code>	wstring class
<code>dtBinary</code>	binary object

Definition at line 65 of file plcBase.h.

7.7.4 Function Documentation**7.7.4.1 reset_and_read() [1/3]**

```
template<typename T , typename U >
bool plc::reset_and_read (
    DataValueTypeDef::atomic\_bool & dirty,
    T & dest,
    U source )
```

Reset and read.

Will read the value and reset the dirty flag.

Definition at line 21 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory_order.

7.7.4.2 reset_and_read() [2/3]

```
template<typename T >
bool plc::reset_and_read (
    DataValueTypeDef::atomic_bool & dirty,
    T & dest,
    typename DataValueTraits< T >::traits_atomic * source )
```

Reset and read.

Will read the value and reset the dirt flag (same type).

Definition at line 34 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory_order.

7.7.4.3 reset_and_read() [3/3]

```
template<>
bool plc::reset_and_read (
    DataValueTypeDef::atomic_bool & dirty,
    DataValueTypeDef::type_wstring & dest,
    DataValueTypeDef::atomic_string * source )
```

Reset and read.

Will read the value and reset the dirty flag (wstring from string).

Definition at line 58 of file plcBase.cpp.

References plc::DataValueTypeDef::memory_order.

Referenced by plc::DataValue::Read().

Here is the caller graph for this function:



7.7.4.4 ScannerProc()

```
VOID CALLBACK plc::ScannerProc (
    LPVOID lpArg,
    DWORD dwTimerLowValue,
    DWORD dwTimerHighValue )
```

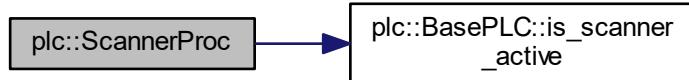
Scanner thread callback with periodic timer

Definition at line 657 of file plcBase.cpp.

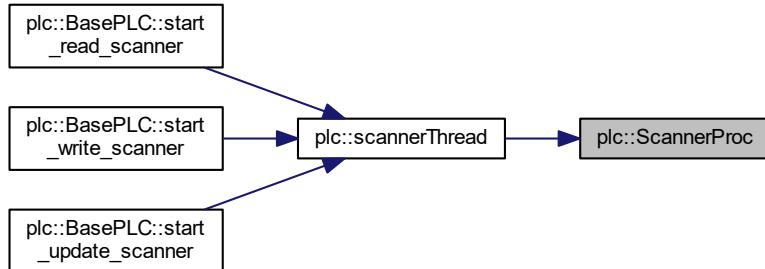
References `plc::BasePLC::is_scanner_active()`, `plc::scanner_thread_args::plc`, and `plc::scanner_thread_args::scanner`.

Referenced by `scannerThread()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.4.5 scannerThread()

```
DWORD WINAPI plc::scannerThread (
    scanner_thread_args args )
```

Scanner thread with periodic timer This function uses the windows waitable timer which will call a completion routine at a regular interval. The completion routine in this case is one of either read_scanner, write_scanner, or update_scanner.

Definition at line 675 of file plcBase.cpp.

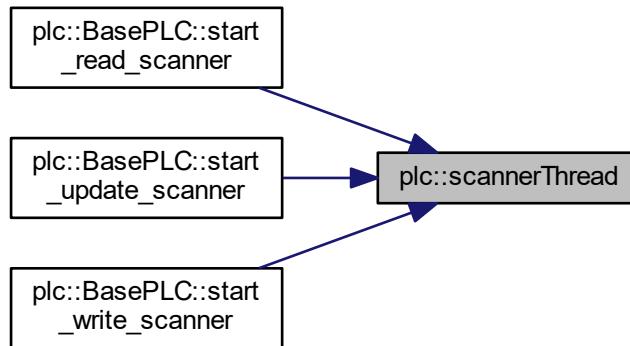
References ScannerProc(), and plc::scanner_thread_args::scanperiod.

Referenced by plc::BasePLC::start_read_scanner(), plc::BasePLC::start_update_scanner(), and plc::BasePLC::start_write_scanner().

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.4.6 write_and_test() [1/3]

```
template<typename T , typename U >
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    U dest,
    const T & source )
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one.

Definition at line 48 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory_order.

7.7.4.7 write_and_test() [2/3]

```
template<typename T >
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    typename DataValueTraits< T >::traits_atomic * dest,
    const T & source )
```

Write and test.

Will set the dirty bit, when the newly written value is different from the old one (same type).

Definition at line 68 of file plcBaseTemplate.h.

References plc::DataValueTypeDef::memory_order.

7.7.4.8 write_and_test() [3/3]

```
template<>
bool plc::write_and_test (
    DataValueTypeDef::atomic_bool & dirty,
    const DataValueTypeDef::atomic_bool & read_pending,
    DataValueTypeDef::atomic_bool & valid,
    DataValueTypeDef::atomic_wstring * dest,
    const DataValueTypeDef::type_string & source )
```

Write and test.

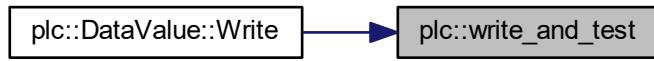
Will set the dirty bit, when the newly written value is different from the old one (string to wstring).

Definition at line 75 of file plcBase.cpp.

References `plc::DataValueTypeDef::memory_order`.

Referenced by `plc::DataValue::Write()`.

Here is the caller graph for this function:



7.8 TcComms Namespace Reference

Namespace for TCat communication.

Classes

- class [AmsRouterNotification](#)

AMS Router Notification.

- struct [DataPar](#)

Memory location struct.

- class [TCatInterface](#)

TCat interface class.

- class [TcPLC](#)

TwinCAT PLC.

- class [tcProcWrite](#)

TwinCAT process write requests.

Functions

- void [errorPrintf](#) (int nErr)

errorPrintf

- bool [compByOffset](#) (BaseRecordPtr recA, BaseRecordPtr recB)

- void __stdcall [ADScallback](#) (AmsAddr *pAddr, AdsNotificationHeader *pNotification, unsigned long plclId)

- void __stdcall [RouterCall](#) (long nReason)

Variables

- const int **MAX_REQ_SIZE** = 250000
maximum allowed request size (bytes)
- const int **MAX_SINGLE_GAP_SIZE** = 50
maximum allowed size (bytes) of a memory gap within continuous request
- const double **MAX_REL_GAP** = 0.25
(maximum allowed total gap size) / (current request size)
- const int **MIN_REL_GAP_SIZE** = 100
minimum allowed relative gap size (bytes)
- const int **default_scanrate** = 100
default PLC TwinCAT scan rate (100ms)
- const int **minimum_scanrate** = 5
minimum PLC TwinCAT scan rate (5ms)
- const int **maximum_scanrate** = 10000
maximum PLC TwinCAT scan rate (10s)
- const int **default_multiple** = 10
default multiple for PLC EPICS scan rate (10)
- const int **minimum_multiple** = 1
minimum multiple for PLC EPICS scan rate (1)
- const int **maximum_multiple** = 200
maximum multiple for PLC EPICS scan rate (200)

7.8.1 Detailed Description

Namespace for TCat communication.

[TcComms](#) name space, which has all the classes and functions used for communicating with TCat.

7.8.2 Function Documentation

7.8.2.1 ADScallback()

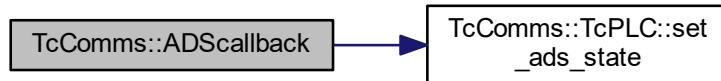
```
void __stdcall TcComms::ADScallback (
    AmsAddr * pAddr,
    AdsNotificationHeader * pNotification,
    unsigned long plcId )
```

Callback for ADS state change

Definition at line 545 of file tcComms.cpp.

References [TcComms::TcPLC::set_ads_state\(\)](#).

Here is the call graph for this function:



7.8.2.2 compByOffset()

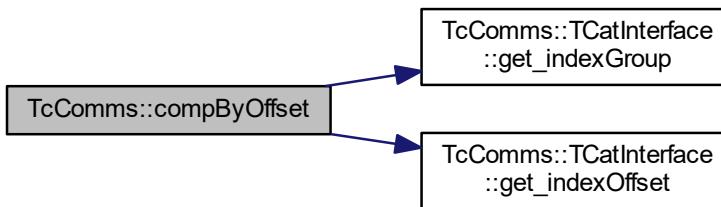
```
bool TcComms::compByOffset (
    BaseRecordPtr recA,
    BaseRecordPtr recB )
```

Compare two TCat records by their group and offset: compbyOffset

Definition at line 377 of file tcComms.cpp.

References TcComms::TCatInterface::get_indexGroup(), and TcComms::TCatInterface::get_indexOffset().

Here is the call graph for this function:



7.8.2.3 errorPrintf()

```
void TcComms::errorPrintf (
    int nErr )
```

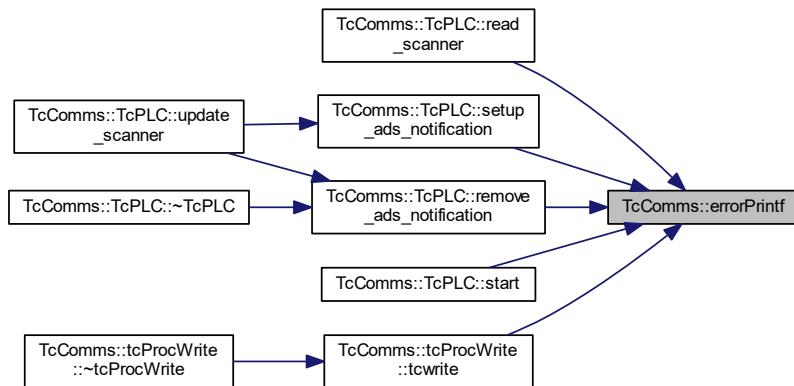
errorPrintf

Print an error message for an ADS error return code

Definition at line 30 of file tcComms.cpp.

Referenced by TcComms::TcPLC::read_scanner(), TcComms::TcPLC::remove_ads_notification(), TcComms::TcPLC::setup_ads_notification(), TcComms::TcPLC::start(), and TcComms::tcProcWrite::tcwrite().

Here is the caller graph for this function:



7.8.2.4 RouterCall()

```
void __stdcall TcComms::RouterCall (
    long nReason )
```

Callback for AMS router state change

Definition at line 757 of file tcComms.cpp.

Chapter 8

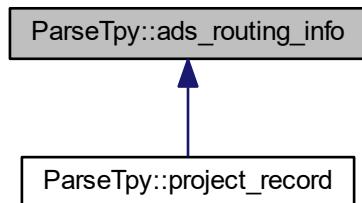
Class Documentation

8.1 ParseTpy::ads_routing_info Class Reference

ADS routing information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::ads_routing_info:



Public Member Functions

- `ads_routing_info ()`
Default constructor.
- `ads_routing_info (const std::string& netid, int port=801)`
Constructor.
- `ads_routing_info (const std::string& netid, int port, const std::string& targetname)`
Constructor.
- `const std::string& get_netid () const`
Get ADS net id.
- `void set_netid (const std::string& netid)`
Set ADS net id.
- `int get_port () const`

- `void set_port (int port)`
Set ADS port.
- `const std::stringcase & get_targetname () const`
Get ADS target name.
- `void set_targetname (const std::stringcase &targetname)`
Set ADS target name.
- `bool isValid () const`
Checks, if net id is of the form n.n.n.n.n.n.
- `std::stringcase get () const`
- `bool set (const std::stringcase &s)`
- `bool get (unsigned char &a1, unsigned char &a2, unsigned char &a3, unsigned char &a4, unsigned char &a5, unsigned char &a6) const`

Protected Attributes

- `std::stringcase ads_netid`
ADS net ID.
- `int ads_port`
ADS port.
- `std::stringcase ads_targetname`
ADS target name.

8.1.1 Detailed Description

ADS routing information.

This is a base class for storing the ADS routing information

Definition at line 23 of file ParseTpy.h.

8.1.2 Member Function Documentation

8.1.2.1 `get()` [1/2]

```
std::stringcase ParseTpy::ads_routing_info::get ( ) const
```

Gets a string representation of a ads routing information

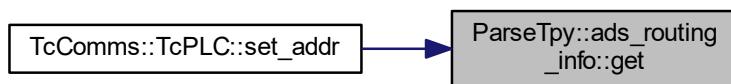
Returns

string with format "tc://netid:port/", empty on error

Definition at line 214 of file ParseTpy.cpp.

Referenced by TcComms::TcPLC::set_addr().

Here is the caller graph for this function:

**8.1.2.2 get() [2/2]**

```
bool ParseTpy::ads_routing_info::get (
    unsigned char & a1,
    unsigned char & a2,
    unsigned char & a3,
    unsigned char & a4,
    unsigned char & a5,
    unsigned char & a6 ) const
```

Get address in net format

Parameters

<i>a1</i>	First address qualifier (return)
<i>a2</i>	Second address qualifier (return)
<i>a3</i>	Third address qualifier (return)
<i>a4</i>	Fourth address qualifier (return)
<i>a5</i>	Fifth address qualifier (return)
<i>a6</i>	Sixth address qualifier (return)

Definition at line 227 of file ParseTpy.cpp.

8.1.2.3 set()

```
bool ParseTpy::ads_routing_info::set (
    const std::string& s )
```

Set the ads routing information using a string of the form: "tc://netid:port/" where netid is a string of the format n.n.n.n.n

Parameters

s	String describing the ads routing information
---	---

Returns

True if successful

Definition at line 248 of file ParseTpy.cpp.

The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

8.2 TcComms::AmsRouterNotification Class Reference

AMS Router Notification.

```
#include <tcComms.h>
```

Static Public Member Functions

- static AmsRouterEvent [get_router_notification \(\)](#)
get router notification

Friends

- void __stdcall [RouterCall \(long\)](#)
Notification callback is a friend.

8.2.1 Detailed Description

AMS Router Notification.

Class for a AMS router notifications

Definition at line 397 of file tcComms.h.

8.2.2 Friends And Related Function Documentation

8.2.2.1 RouterCall

```
void __stdcall RouterCall (
    long nReason ) [friend]
```

Notification callback is a friend.

Callback for AMS router state change

Definition at line 757 of file tcComms.cpp.

The documentation for this class was generated from the following files:

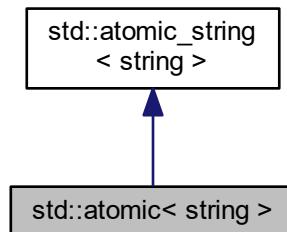
- [tcComms.h](#)
- [tcComms.cpp](#)

8.3 std::atomic< string > Class Template Reference

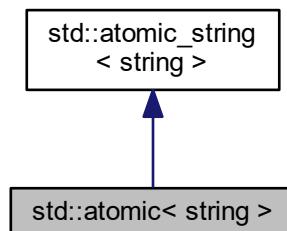
[atomic<string>](#)

```
#include <atomic_string.h>
```

Inheritance diagram for std::atomic< string >:



Collaboration diagram for std::atomic< string >:



Public Member Functions

- [atomic \(\)](#)
Default constructor.
- [atomic \(const string &s\)](#)
Constructor from string.

Additional Inherited Members

8.3.1 Detailed Description

```
template<>
class std::atomic< string >
```

[atomic<string>](#)

This is a class implements an atomic specialization for std::string

Definition at line 71 of file atomic_string.h.

The documentation for this class was generated from the following file:

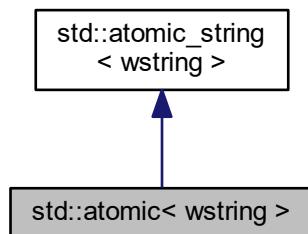
- [atomic_string.h](#)

8.4 std::atomic< wstring > Class Template Reference

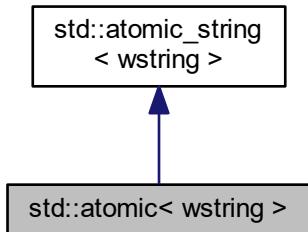
[atomic<wstring>](#)

```
#include <atomic_string.h>
```

Inheritance diagram for std::atomic< wstring >:



Collaboration diagram for std::atomic< wstring >:



Public Member Functions

- [atomic \(\)](#)
Default constructor.
- [atomic \(const wstring &s\)](#)
Constructor from string.

Additional Inherited Members

8.4.1 Detailed Description

```
template<>
class std::atomic< wstring >
```

[atomic<wstring>](#)

This is a class implements an atomic specialization for std::wstring

Definition at line 84 of file atomic_string.h.

The documentation for this class was generated from the following file:

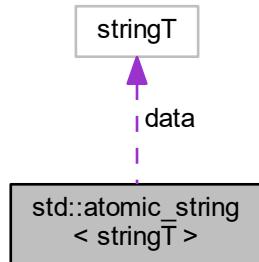
- [atomic_string.h](#)

8.5 std::atomic_string< stringT > Class Template Reference

atomic strings

```
#include <atomic_string.h>
```

Collaboration diagram for std::atomic_string< stringT >:



Public Member Functions

- [atomic_string \(\)](#)
Default constructor.
- [atomic_string \(const stringT &s\)](#)
Constructor from data.
- [stringT operator= \(const stringT &right\)](#)
Assignment operator on basic type.
- [bool is_lock_free \(\) const](#)
Not lock free.
- [void store \(stringT value, memory_order order=memory_order_seq_cst\)](#)
Store.
- [stringT load \(memory_order order=memory_order_seq_cst\) const](#)
Load.
- [operator stringT \(\) const](#)
Convert to string.
- [stringT exchange \(const stringT &value, memory_order order=memory_order_seq_cst\)](#)
Exchange.
- [bool compare_exchange_weak \(stringT &exp, const stringT &value, memory_order order1, memory_order order2\)](#)
Compare exchange.
- [bool compare_exchange_weak \(stringT &exp, const stringT &value, memory_order order=memory_order_seq_cst\)](#)
Compare exchange.
- [bool compare_exchange_strong \(stringT &exp, const stringT &value, memory_order order1, memory_order order2\)](#)
Compare exchange.
- [bool compare_exchange_strong \(stringT &exp, const stringT &value, memory_order order=memory_order_seq_cst\)](#)
Compare exchange.

Protected Attributes

- atomic_flag [flag](#)
flag for spin lock
- stringT [data](#)
data string

8.5.1 Detailed Description

```
template<typename stringT>
class std::atomic_string< stringT >
```

atomic strings

This is a class implements an atomic specialization for strings

Definition at line 16 of file atomic_string.h.

8.5.2 Member Function Documentation

8.5.2.1 operator=()

```
template<typename stringT>
stringT std::atomic_string< stringT >::operator= (
    const stringT & right )
```

Assignment operator on basic type.

Assignment operator.

Definition at line 96 of file atomic_string.h.

The documentation for this class was generated from the following file:

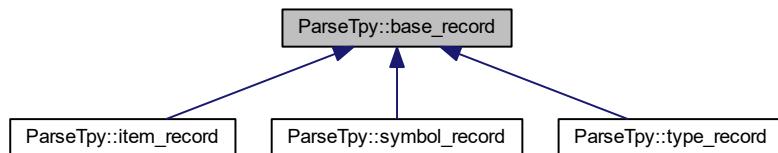
-
- [atomic_string.h](#)

8.6 ParseTpy::base_record Class Reference

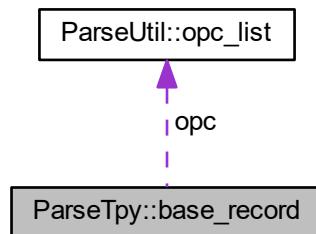
Base record definition.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::base_record:



Collaboration diagram for ParseTpy::base_record:



Public Member Functions

- **base_record ()**
Default constructor.
- **base_record (const std::stringcase &n)**
- **base_record (const std::stringcase &n, const ParseUtil::opc_list &o)**
- **base_record (const std::stringcase &n, const ParseUtil::opc_list &o, const std::stringcase &tn, int td=0)**
- **base_record (const std::stringcase &n, const std::stringcase &tn, int td=0)**
- **const std::stringcase & get_name () const**
Get name.
- **std::stringcase & get_name ()**
Get name.
- **void set_name (std::stringcase n)**
Set name.
- **const std::stringcase & get_type_name () const**
Get type name.

- `std::stringcase & get_type_name ()`
Get type name.
- `void set_type_name (std::stringcase t)`
Set type name.
- `unsigned int get_type_decoration () const`
Get type decoration.
- `void set_type_decoration (unsigned int id)`
Set type decoration.
- `bool get_type_pointer () const`
Get type pointer.
- `void set_type_pointer (bool isPointer)`
Set type pointer.
- `const ParseUtil::opc_list & get_opc () const`
Get OPC list.
- `ParseUtil::opc_list & get_opc ()`
Get OPC list.

Protected Attributes

- `std::stringcase name`
name of type
- `std::stringcase type_n`
type definition
- `unsigned int type_decoration`
decoration or type ID of type definition
- `bool type_pointer`
this is a pointer
- `ParseUtil::opc_list opc`
list of opc properties

8.6.1 Detailed Description

Base record definition.

This is a base class for storing name, type, type id and opc list

Definition at line 145 of file ParseTpy.h.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 base_record() [1/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n ) [inline], [explicit]
```

Constructor

Parameters

<i>n</i>	Name
----------	------

Definition at line 152 of file ParseTpy.h.

8.6.2.2 base_record() [2/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n,
    const ParseUtil::opc_list & o ) [inline]
```

Constructor**Parameters**

<i>n</i>	Name
<i>o</i>	OPC list

Definition at line 157 of file ParseTpy.h.

8.6.2.3 base_record() [3/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n,
    const ParseUtil::opc_list & o,
    const std::stringcase & tn,
    int td = 0 ) [inline]
```

Constructor**Parameters**

<i>n</i>	Name
<i>o</i>	OPC list
<i>tn</i>	Type name
<i>td</i>	Type decortation or id

Definition at line 164 of file ParseTpy.h.

8.6.2.4 base_record() [4/4]

```
ParseTpy::base_record::base_record (
    const std::stringcase & n,
```

```
const std::stringcase & tn,
int td = 0 ) [inline]
```

Constructor

Parameters

<i>n</i>	Name
<i>tn</i>	Type name
<i>td</i>	Type decortation or id

Definition at line 171 of file ParseTyp.h.

The documentation for this class was generated from the following file:

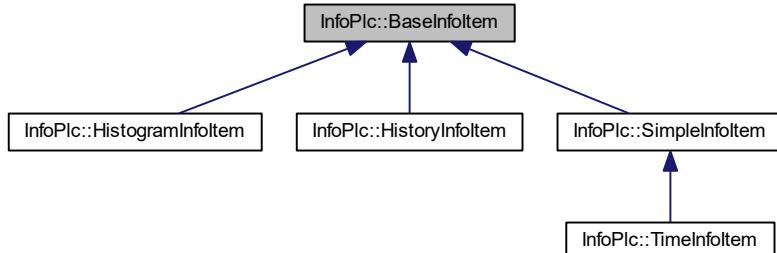
- [ParseTyp.h](#)

8.7 InfoPlc::BaseInfoItem Class Reference

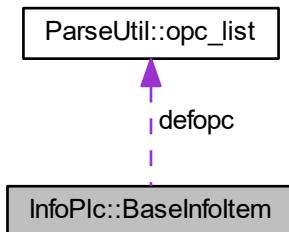
Base info item.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::BaseInfoItem:



Collaboration diagram for InfoPlc::BaseInfoItem:



Public Member Functions

- `BaseInfoItem ()`
Default constructor.
- `BaseInfoItem (const ParseUtil::opc_list &opc)`
Initialize from OPC list.
- `virtual ~BaseInfoItem ()`
Destructor.
- `virtual bool setup (const std::string &tagname, const std::string &plc_alias, plc::Interface *puser)=0`
- `virtual void update (double val)=0`
Update.
- `virtual void reset ()`
Reset.
- `const ParseUtil::opc_list & get_opc () const`
Get OPC list.
- `ParseUtil::opc_list & get_opc ()`
Get OPC list.
- `virtual bool get_info (int idx, const std::string &prefix, std::string &name, ParseUtil::process_type_enum &pctype, ParseUtil::opc_list &opc, std::string &type_n, bool &atomic) const`
get info

Protected Attributes

- `stat_list stats`
statistics list
- `ParseUtil::opc_list defopc`
OPC list.

8.7.1 Detailed Description

Base info item.

This is a base class for a property/value/etc. to keep statistics on

Definition at line 61 of file infoPlc.h.

8.7.2 Member Function Documentation

8.7.2.1 get_info()

```
bool InfoPlc::BaseInfoItem::get_info (
    int idx,
    const std::string& prefix,
    std::string& name,
    ParseUtil::process_type_enum & ptype,
    ParseUtil::opc_list & opc,
    std::string& type_n,
    bool & atomic ) const [virtual]
```

get info

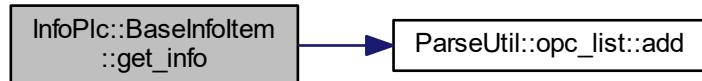
[BaseInfoItem::get_info](#)

Definition at line 15 of file infoPlc.cpp.

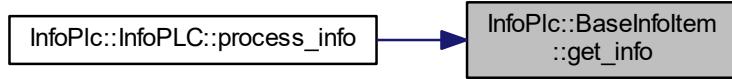
References ParseUtil::opc_list::add(), plc::dtBinary, plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, plc::dtWString, ParseUtil::pt_binary, ParseUtil::pt_bool, ParseUtil::pt_int, ParseUtil::pt_real, and ParseUtil::pt_string.

Referenced by InfoPlc::InfoPLC::process_info().

Here is the call graph for this function:



Here is the caller graph for this function:



8.7.2.2 setup()

```
virtual bool InfoPlc::BaseInfoItem::setup (
    const std::string& tagname,
    const std::string& plc_alias,
    plc::Interface * puser ) [pure virtual]
```

Setup

Parameters

<i>tagname</i>	Name of channels
<i>plc_alias</i>	PLC alias name
<i>puser</i>	Pointer ot PLC

Implemented in [InfoPlc::HistoryInfoItem](#), and [InfoPlc::SimpleInfoItem](#).

The documentation for this class was generated from the following files:

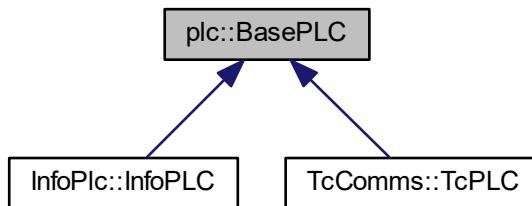
- [infoPlc.h](#)
- [infoPlc.cpp](#)

8.8 plc::BasePLC Class Reference

Base PLC.

```
#include <plcBase.h>
```

Inheritance diagram for plc::BasePLC:



Public Types

- `typedef std::recursive_mutex mutex_type`
Defines the mutex type.
- `typedef std::lock_guard< mutex_type > guard`
Defined the mutex guard type.
- `typedef DataValueTypeDef::type_uint64 time_type`
Define timestamp type.
- `typedef void(BasePLC::* scanner_func) ()`
Function pointer to scanner.

Public Member Functions

- **BasePLC ()**
Default constructor.
- **virtual ~BasePLC ()**
Destructor.
- **int get_read_scanner_period () const**
Get read scanner period in ms.
- **void set_read_scanner_period (int period)**
Set read scanner period in ms.
- **bool start_read_scanner ()**
Start read scanner.
- **bool terminate_read_scanner ()**
Terminate read scanner.
- **int get_write_scanner_period () const**
Get write scanner period in ms.
- **void set_write_scanner_period (int period)**
Set write scanner period in ms.
- **bool start_write_scanner ()**
Start write scanner.
- **bool terminate_write_scanner ()**
Terminate write scanner.
- **int get_update_scanner_period () const**
Get update scanner period in ms.
- **void set_update_scanner_period (int period)**
Set update scanner period in ms.
- **bool start_update_scanner ()**
Start update scanner.
- **bool terminate_update_scanner ()**
Terminate update scanner.
- **bool is_scanner_active () const**
is scanner active?
- **void set_scanners_active (bool active)**
set scanner active state
- **void reserve (BaseRecordList::size_type n)**
- **bool add (BaseRecord *precord)**
- **bool add (BaseRecordPtr precord)**
- **BaseRecordPtr find (const std::string& name)**
- **bool erase (const std::string& name)**
- **bool get_next (BaseRecordPtr &next, const BaseRecordPtr &prev) const**
- template<typename func>
 void for_each (func &f)
- **int count ()**
Count the number of records.
- **virtual void printAllRecords ()**
Print all records and vals to stdout. (override for action)
- **virtual time_type get_timestamp () const**
Get time stamp.
- **virtual time_t get_timestamp_unix () const**
- **virtual void set_timestamp (time_type tstamp)**
Set time stamp.
- **virtual void update_timestamp ()**

- const `std::stringcase & get_name () const`
Get name.
- const `std::stringcase & get_alias () const`
Get nick name/alias.
- void `set_alias (const std::stringcase &nickname)`
Set nick name/alias.
- virtual bool `start ()`
- virtual void `user_data_set_valid (bool valid)`
- virtual void `plc_data_set_valid (bool valid)`

Protected Member Functions

- void `set_name (const std::stringcase &n)`
Set name (careful! This is used for indexing in the PLCList of `System`)
- virtual void `read_scanner ()`
read scanner (override for action)
- virtual void `write_scanner ()`
write scanner (override for action)
- virtual void `update_scanner ()`
update scanner (override for action)

Protected Attributes

- `mutex_type mux`
Mutex to synchronize access to this class.
- `std::stringcase name`
Name.
- `std::stringcase alias`
Nick name or alias (used to generate info record names)
- `BaseRecordList records`
- `time_type timestamp`
Time stamp.
- int `read_scanner_period`
read scanner period in ms
- int `write_scanner_period`
write scanner period in ms
- int `update_scanner_period`
update scanner period in ms
- std::atomic< bool > `scanners_active`
scanners are active
- std::thread `read_thread`
read thread
- std::thread `write_thread`
write thread
- std::thread `update_thread`
update thread

8.8.1 Detailed Description

Base PLC.

This is a base class for interfacing a programmable logic controller. It contains and manages a list of tag/channel records. This is a base class which needs to be used a derived class by a real implementation.

This class is MT safe and uses a mutex to synchronize access.

Definition at line 711 of file plcBase.h.

8.8.2 Member Function Documentation

8.8.2.1 add() [1/2]

```
bool plc::BasePLC::add (
    BaseRecord * precord ) [inline]
```

Add a new tag/channel record. Adding a duplicate is not possible.

Parameters

<i>precord</i>	Pointer to record. Will be adopted
----------------	------------------------------------

Returns

true, if it could be added

Definition at line 775 of file plcBase.h.

8.8.2.2 add() [2/2]

```
bool plc::BasePLC::add (
    BaseRecordPtr precord )
```

Add a new tag/channel record. Adding a duplicate is not possible.

Parameters

<i>precord</i>	Smart pointer to record.
----------------	--------------------------

Returns

true, if it could be added

Definition at line 533 of file plcBase.cpp.

References mux, and records.

8.8.2.3 `erase()`

```
bool plc::BasePLC::erase (
    const std::string& name )
```

Erase a tag/channel record.

Parameters

<i>name</i>	Name of record
-------------	----------------

Returns

true if erased

Definition at line 561 of file plcBase.cpp.

References name, and records.

8.8.2.4 `find()`

```
BaseRecordPtr plc::BasePLC::find (
    const std::string& name )
```

Find a new tag/channel record.

Parameters

<i>name</i>	Name of record
-------------	----------------

Returns

Smart pointer to record (contains nullptr when not found)

Definition at line 547 of file plcBase.cpp.

References mux, name, and records.

8.8.2.5 for_each()

```
template<typename func >
void plc::BasePLC::for_each (
    func & f )
```

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

Parameters

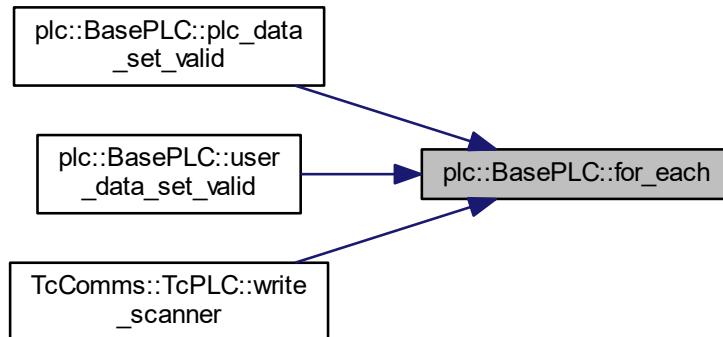
<i>f</i>	Function which takes BaseRecord* as the argument
----------	--

Definition at line 205 of file plcBaseTemplate.h.

References mux, and records.

Referenced by plc_data_set_valid(), user_data_set_valid(), and TcComms::TcPLC::write_scanner().

Here is the caller graph for this function:



8.8.2.6 get_next()

```
bool plc::BasePLC::get_next (
    BaseRecordPtr & next,
    const BaseRecordPtr & prev ) const
```

Get next record in list. Resets to the beginning, if the list changes between subsequent access. This is an MT safe access method to cycle through the list. However, it is not high performance and is meant for slow external access.

Parameters

<i>next</i>	Next tag/channel record (return)
<i>prev</i>	tag/channel record

Returns

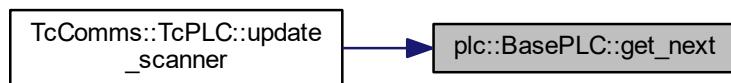
true if successful

Definition at line 569 of file plcBase.cpp.

References mux, and records.

Referenced by TcComms::TcPLC::update_scanner().

Here is the caller graph for this function:

**8.8.2.7 get_timestamp_unix()**

```
time_t plc::BasePLC::get_timestamp_unix ( ) const [virtual]
```

Get time stamp as unix time (seconds since 1970-01-01 00:00:00) Does not include leap seconds

Definition at line 593 of file plcBase.cpp.

References timestamp.

8.8.2.8 plc_data_set_valid()

```
void plc::BasePLC::plc_data_set_valid (
    bool valid ) [virtual]
```

Set the valid flag for all data values by the plc

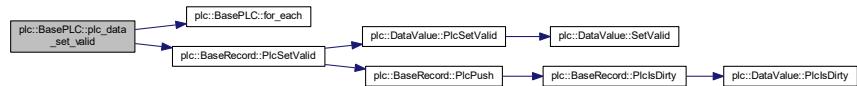
Parameters

<i>valid</i>	Valid flag, true for valid, false for invalid
--------------	---

Definition at line 635 of file plcBase.cpp.

References for_each(), and plc::BaseRecord::PlcSetValid().

Here is the call graph for this function:



8.8.2.9 reserve()

```
void plc::BasePLC::reserve (
    BaseRecordList::size_type n ) [inline]
```

Reserves the given number of elements in the tag/channel list. Use this function when you know many elements are added beforehand to avoid unnecessary rehashing.

Parameters

<i>n</i>	Number of expected tag/channel records
----------	--

Definition at line 770 of file plcBase.h.

References records.

8.8.2.10 start()

```
virtual bool plc::BasePLC::start () [inline], [virtual]
```

This function is called by the driver support in [drvtc.cpp](#), after the PLC has been initialized. This function is overridden by the derived PLCs, and generally will start all the scanner threads.

Returns

true if successful

Reimplemented in [TcComms::TcPLC](#).

Definition at line 831 of file plcBase.h.

8.8.2.11 user_data_set_valid()

```
void plc::BasePLC::user_data_set_valid (
    bool valid ) [virtual]
```

Set the valid flag for all data values by the user

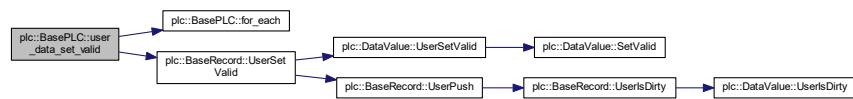
Parameters

<code>valid</code>	Valid flag, true for valid, false for invalid
--------------------	---

Definition at line 625 of file plcBase.cpp.

References for_each(), and `plc::BaseRecord::UserSetValid()`.

Here is the call graph for this function:



8.8.3 Member Data Documentation

8.8.3.1 records

`BaseRecordList plc::BasePLC::records [protected]`

List of tags/channels. The load factor is initialized to 0.5.

Definition at line 852 of file plcBase.h.

Referenced by add(), BasePLC(), count(), erase(), find(), for_each(), get_next(), TcComms::TcPLC::printAllRecords(), TcComms::TcPLC::read_scanner(), reserve(), and TcComms::TcPLC::update_scanner().

The documentation for this class was generated from the following files:

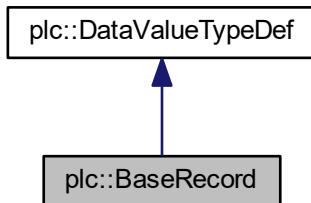
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.9 plc::BaseRecord Class Reference

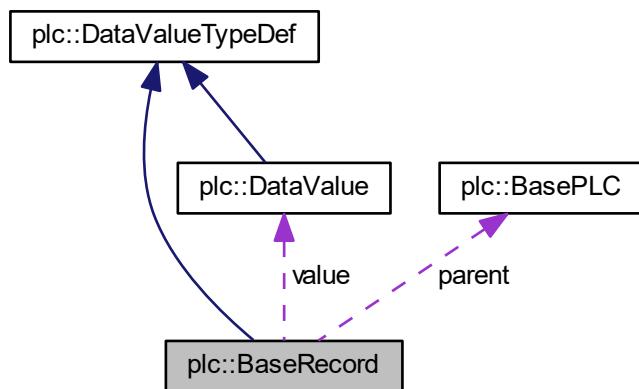
Class for managing a tag/channel.

```
#include <plcBase.h>
```

Inheritance diagram for plc::BaseRecord:



Collaboration diagram for plc::BaseRecord:



Public Member Functions

- [BaseRecord \(\)](#)
Default constructor.
- [BaseRecord \(const std::string& tag\)](#)
- [BaseRecord \(const std::string& recordName, data_type_enum rt, Interface *puser=nullptr, Interface *pplc=nullptr\)](#)
- [virtual ~BaseRecord \(\)](#)
Destructor.

- const std::stringcase & **get_name** () const
Get name.
- void **set_name** (const std::stringcase &recordName)
Set name.
- bool **get_process** () const
Get process flag: false = disabled, true = enabled.
- void **set_process** (bool isEnabled)
Set process flag.
- **access_rights_enum get_access_rights** ()
Get access rights.
- void **set_access_rights** (access_rights_enum rights)
Set access rights.
- **time_type get_timestamp** () const
Get time stamp.
- virtual Interface * **get_userInterface** () const
Get pointer to user interface (no ownership transfer)
- virtual Interface * **get_plcInterface** () const
Get pointer to plc interface (no ownership transfer)
- void **set_userInterface** (Interface *puser)
Set user interface (object will be adopted!)
- void **set_plcInterface** (Interface *pplc)
Set plc interface (object will be adopted!)
- virtual BasePLC * **get_parent** () const
Get parent plc.
- virtual void **set_parent** (BasePLC *pPLC)
Set parent plc.
- const **DataValue & get_data** () const
Get a const reference to the data object.
- **DataValue & get_data** ()
Get a reference to the data object.
- bool **DatasValid** ()
Returns true, if the data is valid.
- template<typename T >
 bool **UserRead** (T &data)
- bool **UserRead** (type_string_value *data, size_type max)
- bool **UserRead** (DataValue::type_wstring_value *data, size_type max)
- template<typename T >
 bool **UserWrite** (const T &data)
- bool **UserWrite** (const type_string_value *data, size_type max)
- bool **UserWrite** (const type_wstring_value *data, size_type max)
- size_type **UserReadBinary** (type_binary p, size_type len)
- size_type **UserWriteBinary** (const type_binary p, size_type len)
- bool **UserIsDirty** () const
Checks if the user needs to read an updated value.
- void **UserSetDirty** ()
Set dirty flag for user.
- bool **UserPush** (bool force=false)
- bool **UserPull** ()
Pulls the user for new data.
- void **UserSetValid** (bool valid)
- bool **UserGetValid** ()

- template<typename T >
bool **PlcRead** (T &data)
- bool **PlcRead** (type_string_value *data, size_type max)
- bool **PlcRead** (type_wstring_value *data, size_type max)
- template<typename T >
bool **PlcWrite** (const T &data)
- bool **PlcWrite** (const type_string_value *data, size_type max)
- bool **PlcWrite** (const type_wstring_value *data, size_type max)
- size_type **PlcReadBinary** (type_binary p, size_type len)
- size_type **PlcWriteBinary** (const type_binary p, size_type len)
- bool **PlcIsDirty** () const
Checks if the plc needs to read an updated value.
- void **PlcSetDirty** ()
Set dirty flag for plc.
- bool **PlcPush** (bool force=false)
- bool **PlcPull** ()
Pulls the plc for new data.
- void **PlcSetValid** (bool valid)
- bool **PlcGetValid** ()

Protected Attributes

- std::stringcase name
Name.
- access_rights_enum access
Enum for access rights.
- atomic_bool process
Process flag: false = disabled, true = enabled.
- **DataValue** value
Data value.
- InterfacePtr plc
PLC interface (master)
- InterfacePtr user
User interface (slave)
- BasePLC * parent
PLC that this record belongs to.

Additional Inherited Members

8.9.1 Detailed Description

Class for managing a tag/channel.

This is the base class for a tag/channel. It contains a data value and owns two pointers to a user and plc interface, respectively. This is the basic work horse for exchanging data between a user (slave) and a plc (master).

Definition at line 486 of file plcBase.h.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 BaseRecord() [1/2]

```
plc::BaseRecord::BaseRecord (
    const std::string& tag ) [inline], [explicit]
```

Constructor

Parameters

<i>tag</i>	Name of tag/channel
------------	---------------------

Definition at line 493 of file plcBase.h.

8.9.2.2 BaseRecord() [2/2]

```
plc::BaseRecord::BaseRecord (
    const std::string& recordName,
    data_type_enum rt,
    Interface * puser = nullptr,
    Interface * pplc = nullptr ) [inline]
```

Constructor

Parameters

<i>recordName</i>	Name of tag/channel
<i>rt</i>	Data type
<i>puser</i>	Pointer to user interface object (will be adopted!)
<i>pplc</i>	Pointer to plc interface object (will be adopted!)

Definition at line 500 of file plcBase.h.

8.9.3 Member Function Documentation

8.9.3.1 PlcGetValid()

```
bool plc::BaseRecord::PlcGetValid () [inline]
```

Get the plc valid flag and reset the dirty flag

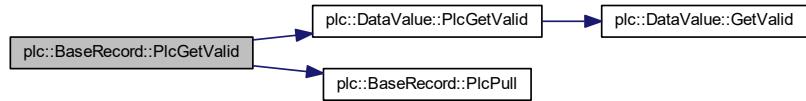
Returns

valid True for valid data, False for invalid

Definition at line 673 of file plcBase.h.

References plc::DataValue::PlcGetValid(), PlcPull(), process, and value.

Here is the call graph for this function:

**8.9.3.2 PlcPush()**

```
bool plc::BaseRecord::PlcPush (
    bool force = false ) [inline]
```

Initiated a plc pull, if the data needs an update

Parameters

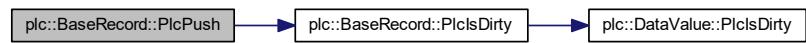
<i>force</i>	Forces a plc update even when not needed
--------------	--

Definition at line 179 of file plcBaseTemplate.h.

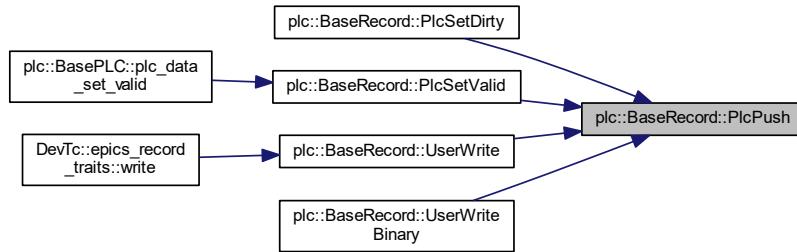
References PlcIsDirty().

Referenced by PlcSetDirty(), PlcSetValid(), UserWrite(), and UserWriteBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.3.3 PlcRead() [1/3]

```
template<typename T >
bool plc::BaseRecord::PlcRead (
    T & data ) [inline]
```

Execute a plc read, but pull user first

Parameters

<code>data</code>	Reference to data (return)
-------------------	----------------------------

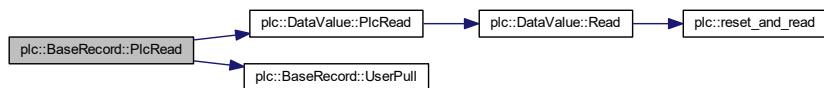
Returns

true if successfull

Definition at line 612 of file `plcBase.h`.

References `plc::DataValue::PlcRead()`, `UserPull()`, and `value`.

Here is the call graph for this function:



8.9.3.4 PlcRead() [2/3]

```
bool plc::BaseRecord::PlcRead (
```

<pre> type_string_value * data,</pre>	<pre> size_type max) [inline]</pre>
--	---

Execute a plc read, but pull user first

Parameters

<i>data</i>	character pointer, pchar (return)
<i>max</i>	Maximum number of characters

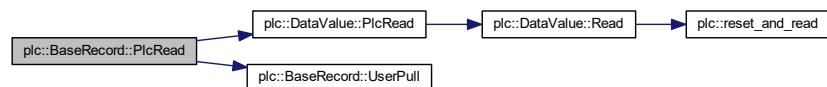
Returns

true if successfull

Definition at line 618 of file plcBase.h.

References `plc::DataValue::PlcRead()`, `UserPull()`, and `value`.

Here is the call graph for this function:

**8.9.3.5 PlcRead() [3/3]**

```
bool plc::BaseRecord::PlcRead (
    type_wstring_value * data,
    size_type max ) [inline]
```

Execute a plc read, but pull user first

Parameters

<i>data</i>	character pointer, pwchar (return)
<i>max</i>	Maximum number of characters

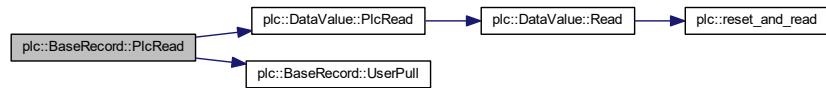
Returns

true if successfull

Definition at line 624 of file plcBase.h.

References `plc::DataValue::PlcRead()`, `UserPull()`, and `value`.

Here is the call graph for this function:



8.9.3.6 PlcReadBinary()

```
size_type plc::BaseRecord::PlcReadBinary (
    type_binary p,
    size_type len ) [inline]
```

Execute a plc read, but pull user first

Parameters

<i>p</i>	Pointer to data (destination buffer)
<i>len</i>	Length in bytes (must be the same as data length)

Returns

Number of bytes read (0 on error)

Definition at line 649 of file plcBase.h.

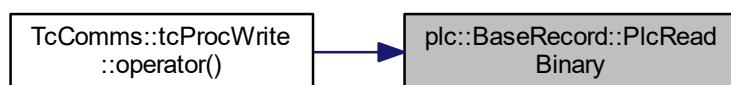
References plc::DataValue::PlcReadBinary(), UserPull(), and value.

Referenced by TcComms::tcProcWrite::operator()().

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.3.7 PlcSetValid()

```
void plc::BaseRecord::PlcSetValid (
    bool valid ) [inline]
```

Set the plc valid flag and set the dirty flag when flag changes

Parameters

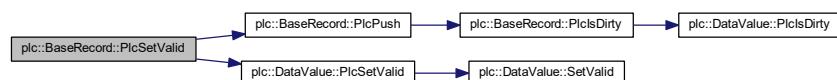
<code>valid</code>	True for valid data, False for invalid
--------------------	--

Definition at line 670 of file plcBase.h.

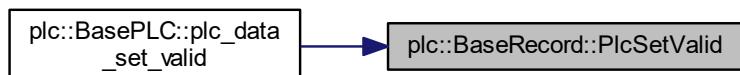
References PlcPush(), plc::DataValue::PlcSetValid(), and value.

Referenced by plc::BasePLC::plc_data_set_valid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.3.8 PlcWrite() [1/3]

```
template<typename T >
bool plc::BaseRecord::PlcWrite (
    const T & data ) [inline]
```

Execute a plc write and push user

Parameters

<code>data</code>	Reference to data
-------------------	-------------------

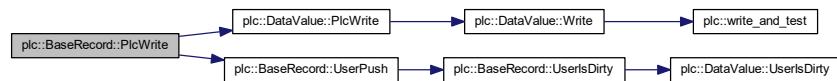
Returns

true if successfull

Definition at line 630 of file plcBase.h.

References `plc::DataValue::PlcWrite()`, `UserPush()`, and `value`.

Here is the call graph for this function:

**8.9.3.9 PlcWrite() [2/3]**

```
bool plc::BaseRecord::PlcWrite (
    const type_string_value * data,
    size_type max ) [inline]
```

Execute a plc write and push user

Parameters

<code>data</code>	character pointer, pchar
<code>max</code>	Maximum number of characters

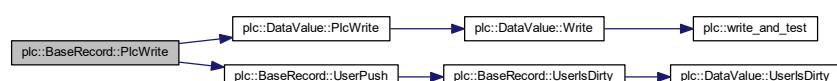
Returns

true if successfull

Definition at line 636 of file plcBase.h.

References `plc::DataValue::PlcWrite()`, `UserPush()`, and `value`.

Here is the call graph for this function:



8.9.3.10 PlcWrite() [3/3]

```
bool plc::BaseRecord::PlcWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Execute a plc write and push user

Parameters

<i>data</i>	character pointer, pwchar
<i>max</i>	Maximum number of characters

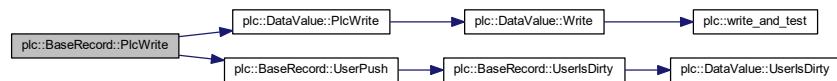
Returns

true if successfull

Definition at line 642 of file plcBase.h.

References `plc::DataValue::PlcWrite()`, `UserPush()`, and `value`.

Here is the call graph for this function:



8.9.3.11 PlcWriteBinary()

```
size_type plc::BaseRecord::PlcWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Execute a plc write and push user

Parameters

<i>p</i>	Pointer to data (source buffer)
<i>len</i>	Length in bytes (must be the same as data length)

Returns

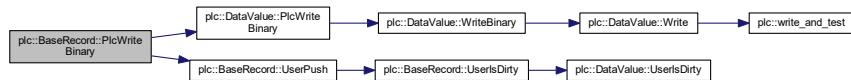
Number of bytes written (0 on error)

Definition at line 655 of file plcBase.h.

References `plc::DataValue::PlcWriteBinary()`, `UserPush()`, and `value`.

Referenced by `TcComms::TcPLC::read_scanner()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.3.12 UserGetValid()

`bool plc::BaseRecord::UserGetValid() [inline]`

Get the user valid flag and reset the dirty flag

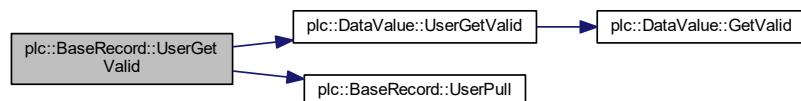
Returns

valid True for valid data, False for invalid

Definition at line 606 of file `plcBase.h`.

References `process`, `plc::DataValue::UserGetValid()`, `UserPull()`, and `value`.

Here is the call graph for this function:



8.9.3.13 UserPush()

`bool plc::BaseRecord::UserPush(bool force = false) [inline]`

Initiated a user pull, if the data needs an update

Parameters

<i>force</i>	Forces a user update even when not needed
--------------	---

Definition at line 153 of file plcBaseTemplate.h.

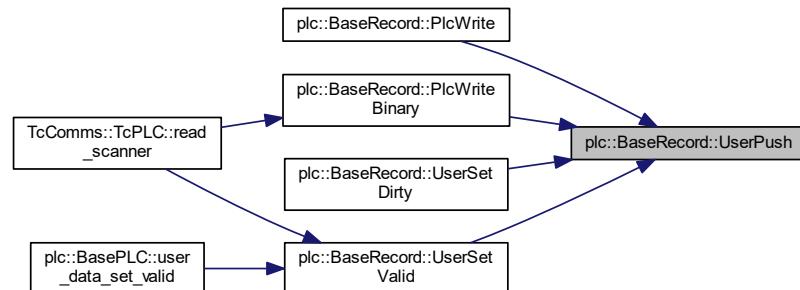
References user, and UserIsDirty().

Referenced by PlcWrite(), PlcWriteBinary(), UserSetDirty(), and UserSetValid().

Here is the call graph for this function:



Here is the caller graph for this function:

**8.9.3.14 UserRead() [1/3]**

```
template<typename T >
bool plc::BaseRecord::UserRead (
    T & data ) [inline]
```

Execute a user read, but pull plc first

Parameters

<i>data</i>	Reference to data (return)
-------------	----------------------------

Returns

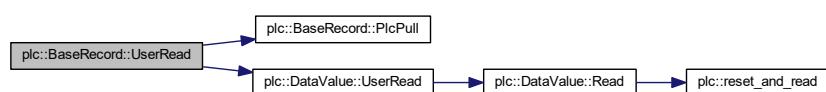
true if successfull

Definition at line 546 of file plcBase.h.

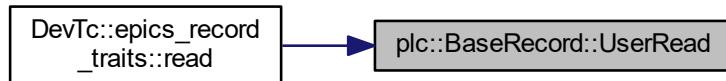
References PlcPull(), plc::DataValue::UserRead(), and value.

Referenced by DevTc::epics_record_traits< RecType >::read().

Here is the call graph for this function:



Here is the caller graph for this function:

**8.9.3.15 UserRead() [2/3]**

```
bool plc::BaseRecord::UserRead (
    type_string_value * data,
    size_type max ) [inline]
```

Execute a user read, but pull plc first

Parameters

<i>data</i>	character pointer, pchar (return)
<i>max</i>	Maximum number of characters

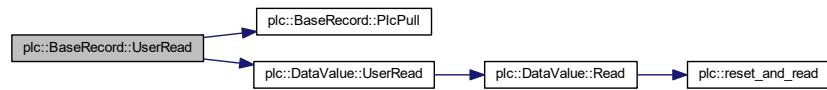
Returns

true if successfull

Definition at line 552 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

Here is the call graph for this function:



8.9.3.16 UserRead() [3/3]

```
bool plc::BaseRecord::UserRead (
    DataValue::type_wstring_value * data,
    size_type max ) [inline]
```

Execute a user read, but pull plc first

Parameters

<i>data</i>	character pointer, pwchar (return)
<i>max</i>	Maximum number of characters

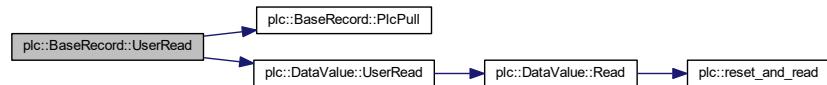
Returns

true if successfull

Definition at line 558 of file plcBase.h.

References PlcPull(), plc::DataValue::UserRead(), and value.

Here is the call graph for this function:



8.9.3.17 UserReadBinary()

```
size_type plc::BaseRecord::UserReadBinary (
    type_binary p,
    size_type len ) [inline]
```

Execute a user read, but pull plc first

Parameters

<i>p</i>	Pointer to data (destination buffer)
<i>len</i>	Length in bytes (must be the same as data length)

Returns

Number of bytes read (0 on error)

Definition at line 582 of file plcBase.h.

References PlcPull(), plc::DataValue::UserReadBinary(), and value.

Here is the call graph for this function:

**8.9.3.18 UserSetValid()**

```
void plc::BaseRecord::UserSetValid (
    bool valid ) [inline]
```

Set the user valid flag and set the dirty flag when flag changes

Parameters

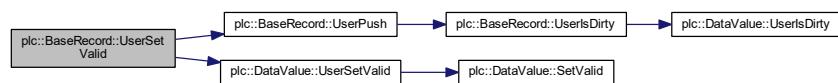
<i>valid</i>	True for valid data, False for invalid
--------------	--

Definition at line 603 of file plcBase.h.

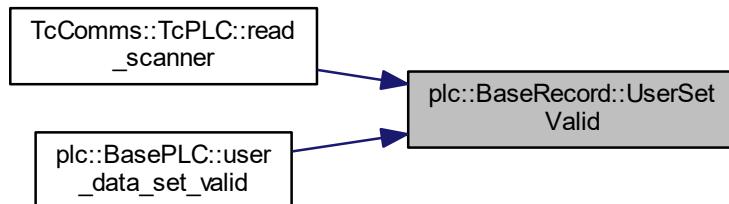
References UserPush(), plc::DataValue::UserSetValid(), and value.

Referenced by TcComms::TcPLC::read_scanner(), and plc::BasePLC::user_data_set_valid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.3.19 UserWrite() [1/3]

```
template<typename T >
bool plc::BaseRecord::UserWrite (
    const T & data )  [inline]
```

Execute a user write and push plc

Parameters

<i>data</i>	Reference to data
-------------	-------------------

Returns

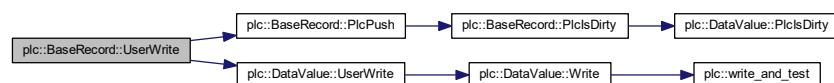
true if successfull

Definition at line 563 of file plcBase.h.

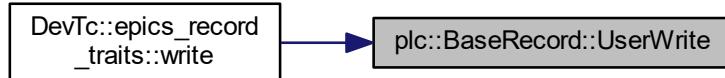
References PlcPush(), plc::DataValue::UserWrite(), and value.

Referenced by DevTc::epics_record_traits< RecType >::write().

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.3.20 UserWrite() [2/3]

```

bool plc::BaseRecord::UserWrite (
    const type_string_value * data,
    size_type max ) [inline]
  
```

Execute a user write and push plc

Parameters

<i>data</i>	character pointer, pchar
<i>max</i>	Maximum number of characters

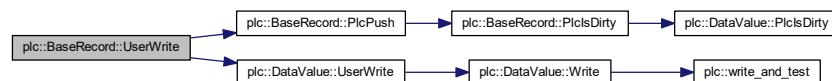
Returns

true if successfull

Definition at line 569 of file plcBase.h.

References PlcPush(), plc::DataValue::UserWrite(), and value.

Here is the call graph for this function:



8.9.3.21 UserWrite() [3/3]

```

bool plc::BaseRecord::UserWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
  
```

Execute a user write and push plc

Parameters

<i>data</i>	character pointer, <code>pwchar</code>
<i>max</i>	Maximum number of characters

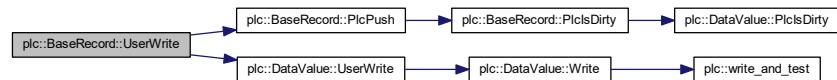
Returns

true if successfull

Definition at line 575 of file `plcBase.h`.

References `PlcPush()`, `plc::DataValue::UserWrite()`, and `value`.

Here is the call graph for this function:

**8.9.3.22 UserWriteBinary()**

```

size_type plc::BaseRecord::UserWriteBinary (
    const type_binary p,
    size_type len ) [inline]
  
```

Execute a user write and push plc

Parameters

<i>p</i>	Pointer to data (source buffer)
<i>len</i>	Length in bytes (must be the same as data length)

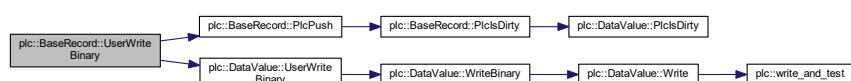
Returns

Number of bytes written (0 on error)

Definition at line 588 of file `plcBase.h`.

References `PlcPush()`, `plc::DataValue::UserWriteBinary()`, and `value`.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

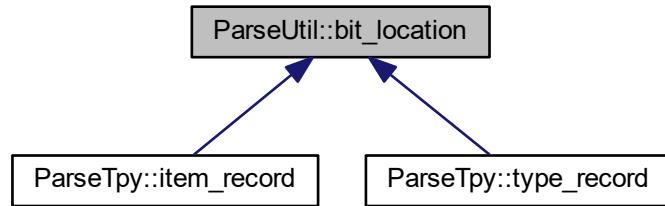
- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.10 ParseUtil::bit_location Class Reference

Bit location.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::bit_location:



Public Member Functions

- [bit_location \(\)](#)
Default constructor.
- [bit_location \(int bo, int bs\)](#)
Constructor.
- [bool isValid \(\) const](#)
Validity.
- [const int get_bit_offset \(\) const](#)
Get bit offset.
- [void set_bit_offset \(int ofs\)](#)
Set bit offset.
- [const int get_bit_size \(\) const](#)
Get bit size.
- [void set_bit_size \(int size\)](#)
Set bit size.

Protected Attributes

- [int bitoffs](#)
bit offset where elements is stored
- [int bitsize](#)
size in number of bits of symbol

8.10.1 Detailed Description

Bit location.

This is a class for storing bit offset and size in a structure

Definition at line 189 of file ParseUtil.h.

The documentation for this class was generated from the following file:

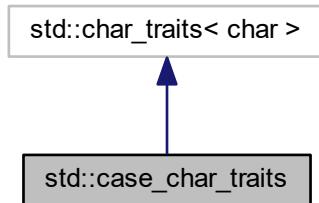
- [ParseUtil.h](#)

8.11 std::case_char_traits Struct Reference

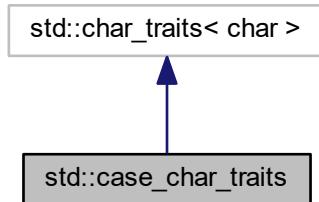
case insensitive traits.

```
#include <stringcase.h>
```

Inheritance diagram for std::case_char_traits:



Collaboration diagram for std::case_char_traits:



Static Public Member Functions

- static bool [eq](#) (const char_type &c1, const char_type &c2)
- static bool [ne](#) (const char_type &c1, const char_type &c2)
- static bool [lt](#) (const char_type &c1, const char_type &c2)
- static int [compare](#) (const char_type *s1, const char_type *s2, size_t n)

8.11.1 Detailed Description

case insensitive traits.

This traits class is not case sensitive.

Definition at line 40 of file stringcase.h.

8.11.2 Member Function Documentation

8.11.2.1 compare()

```
static int std::case_char_traits::compare (
    const char_type * s1,
    const char_type * s2,
    size_t n ) [inline], [static]
```

Compare strings

Parameters

<i>s1</i>	First string
<i>s2</i>	Second string
<i>n</i>	number of characters

Definition at line 64 of file stringcase.h.

References std::strncasecmp().

Here is the call graph for this function:



8.11.2.2 eq()

```
static bool std::case_char_traits::eq (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 45 of file stringcase.h.

8.11.2.3 lt()

```
static bool std::case_char_traits::lt (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Lower than character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 57 of file stringcase.h.

8.11.2.4 ne()

```
static bool std::case_char_traits::ne (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Not equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 51 of file stringcase.h.

The documentation for this struct was generated from the following file:

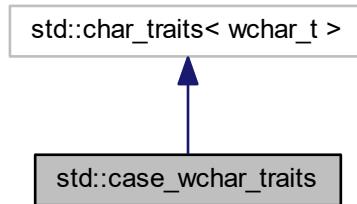
- [stringcase.h](#)

8.12 std::case_wchar_traits Struct Reference

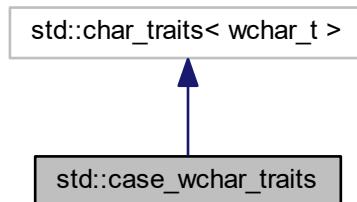
case insensitive unicode traits.

```
#include <stringcase.h>
```

Inheritance diagram for std::case_wchar_traits:



Collaboration diagram for std::case_wchar_traits:



Static Public Member Functions

- static bool [eq](#) (const char_type &c1, const char_type &c2)
- static bool [ne](#) (const char_type &c1, const char_type &c2)
- static bool [lt](#) (const char_type &c1, const char_type &c2)
- static int [compare](#) (const char_type *s1, const char_type *s2, size_t n)

8.12.1 Detailed Description

case insensitive unicode traits.

This unicode traits class is not case sensitive.

Definition at line 72 of file stringcase.h.

8.12.2 Member Function Documentation

8.12.2.1 compare()

```
static int std::case_wchar_traits::compare (
    const char_type * s1,
    const char_type * s2,
    size_t n ) [inline], [static]
```

Compare strings

Parameters

<i>s1</i>	First string
<i>s2</i>	Second string
<i>n</i>	number of characters

Definition at line 96 of file stringcase.h.

References std::wcsncasewcmp().

Here is the call graph for this function:



8.12.2.2 eq()

```
static bool std::case_wchar_traits::eq (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Equal character

Parameters

<i>c1</i>	First char
<i>c2</i>	Second char

Definition at line 77 of file stringcase.h.

8.12.2.3 lt()

```
static bool std::case_wchar_traits::lt (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Lower than character

Parameters

c1	First char
c2	Second char

Definition at line 89 of file stringcase.h.

8.12.2.4 ne()

```
static bool std::case_wchar_traits::ne (
    const char_type & c1,
    const char_type & c2 ) [inline], [static]
```

Not equal character

Parameters

c1	First char
c2	Second char

Definition at line 83 of file stringcase.h.

The documentation for this struct was generated from the following file:

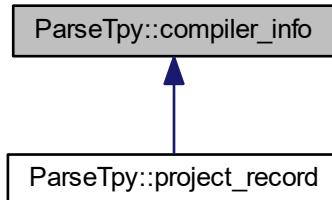
- [stringcase.h](#)

8.13 ParseTpy::compiler_info Class Reference

Compiler information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::compiler_info:



Public Member Functions

- `compiler_info ()`
Default constructor.
- `const std::stringcase & get_cmpl_versionstr () const`
Get compiler version string.
- `void set_cmpl_versionstr (const std::stringcase &versionstr)`
Set compiler version string.
- `double get_cmpl_version () const`
Get compiler version.
- `const std::stringcase & get_tcat_versionstr () const`
Get twincat version string.
- `void set_tcat_versionstr (const std::stringcase &versionstr)`
Set twincat version string.
- `double get_tcat_version () const`
Get twincat version.
- `const std::stringcase & get_cpu_family () const`
Get cpu family string.
- `void set_cpu_family (const std::stringcase &family)`
Set cpu family string.
- `bool is_cmpl_Valid () const`
Checks, if version is of the form n.n...
- `bool is_tcat_Valid () const`
Checks, if twincat version is of the form n.n...

Protected Attributes

- `std::stringcase cmpl_versionstr`
version string
- `double cmpl_version`
version number
- `std::stringcase tcat_versionstr`
twincat version string
- `double tcat_version`
twincat version number
- `std::stringcase cpu_family`
cpu family string

8.13.1 Detailed Description

Compiler information.

This is a base class for storing the compiler information

Definition at line 80 of file ParseTpy.h.

The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

8.14 TcComms::DataPar Struct Reference

Memory location struct.

```
#include <tcComms.h>
```

Public Attributes

- unsigned long **indexGroup**
index group in ADS server
- unsigned long **indexOffset**
index offset in ADS server
- unsigned long **length**
count of bytes to read

8.14.1 Detailed Description

Memory location struct.

Struct for storing index group, index offset, and size of a TC symbol

Definition at line 61 of file tcComms.h.

The documentation for this struct was generated from the following file:

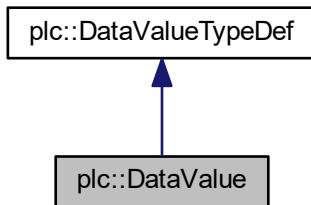
- [tcComms.h](#)

8.15 plc::DataValue Class Reference

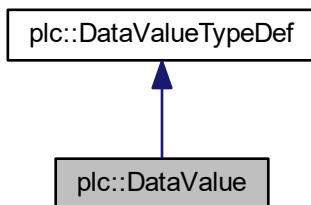
Data value.

```
#include <plcBase.h>
```

Inheritance diagram for plc::DataValue:



Collaboration diagram for plc::DataValue:



Public Types

- `typedef void * data_type`
Internally used storage pointer type.

Public Member Functions

- `DataValue ()`
Default constructor.
- `DataValue (data_type_enum rt, size_type len=0)`
- `~DataValue ()`
Destructor.
- `DataValue (const DataValue &)`

- Copy constructor.*
- `DataValue & operator= (const DataValue &)`
Assignment operator.
 - `void Init (data_type_enum rt, size_type len=0)`
 - `bool IsValid () const`
is valid
 - `data_type_enum get_data_type () const`
get type
 - `size_type get_size () const`
get size
 - template<typename T >
 `bool UserRead (T &data) const`
 - template<size_type N>
 `bool UserRead (type_string_value(&data)[N]) const`
 - `bool UserRead (type_string_value *data, size_type max) const`
 - `bool UserRead (type_wstring_value *data, size_type max) const`
 - template<typename T >
 `bool UserWrite (const T &data)`
 - template<size_type N>
 `bool UserWrite (const type_string_value(&data)[N])`
 - `bool UserWrite (const type_string_value *data, size_type max)`
 - `bool UserWrite (const type_wstring_value *data, size_type max)`
 - `size_type UserReadBinary (type_binary p, size_type len) const`
 - `size_type UserWriteBinary (const type_binary p, size_type len)`
 - `bool UserIsDirty () const`
New data for user.
 - `void UserSetDirty ()`
Set dirty flag for user.
 - `void UserSetValid (bool valid)`
 - `bool UserGetValid () const`
 - template<typename T >
 `bool PlcRead (T &data) const`
 - template<size_type N>
 `bool PlcRead (type_string_value(&data)[N]) const`
 - `bool PlcRead (type_string_value *data, size_type max) const`
 - `bool PlcRead (type_wstring_value *data, size_type max) const`
 - template<typename T >
 `bool PlcWrite (const T &data)`
 - template<size_type N>
 `bool PlcWrite (const type_string_value(&data)[N])`
 - `bool PlcWrite (const type_string_value *data, size_type max)`
 - `bool PlcWrite (const type_wstring_value *data, size_type max)`
 - `size_type PlcReadBinary (type_binary p, size_type len) const`
 - `size_type PlcWriteBinary (const type_binary p, size_type len)`
 - `bool PlcIsDirty () const`
New data for plc.
 - `void PlcSetDirty ()`
Set dirty flag for plc.
 - `void PlcSetValid (bool valid)`
 - `bool PlcGetValid () const`

Protected Member Functions

- `DataValue (const DataValue &&)`
Constructor (hidden)
- `DataValue & operator= (const DataValue &&)`
Assignment operator.
- template<typename T >
`bool Read (atomic_bool &dirty, T &data) const`
- `bool Read (atomic_bool &dirty, type_string &data) const`
- `bool Read (atomic_bool &dirty, type_wstring &data) const`
- `bool Read (atomic_bool &dirty, type_string_value *data, size_type max) const`
- `bool Read (atomic_bool &dirty, type_wstring_value *data, size_type max) const`
- template<typename T >
`bool Write (atomic_bool &dirty, const atomic_bool &pend, const T &data)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_string &data)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_wstring &data)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_string_value *data, size_type max)`
- `bool Write (atomic_bool &dirty, const atomic_bool &pend, const type_wstring_value *data, size_type max)`
- `size_type ReadBinary (atomic_bool &dirty, type_binary p, size_type len) const`
- `size_type WriteBinary (atomic_bool &dirty, const atomic_bool &pend, const type_binary p, size_type len)`
- `void SetValid (atomic_bool &dirty, bool valid)`
- `bool GetValid (atomic_bool &dirty) const`

Protected Attributes

- `data_type mydata`
Data pointer.
- `size_type mysize`
- `data_type_enum mytype`
Data type.
- `atomic_bool myvalid`
Valid flag.
- `atomic_bool myuserdirty`
Dirty flag indicating user needs to update.
- `atomic_bool myplcdirty`
Dirty flag indicating plc needs to update.

Additional Inherited Members

8.15.1 Detailed Description

Data value.

Class for data value This class stores a data value and provides synchronization between the user (slave) and the plc (master) interfaces. When the plc writes a value, it is marked dirty on the user side. Then, when the user reads this data, it resets the dirty flag. The same logic applies for writes by the user and reads by the plc.

Data access is guaranteed to be atomic and MT safe. Construction, initialization and destruction is not MT safe and all data access has to be stopped during these operations.

Type conversion is provided between all simple data types. However, the loss of information is not checked upon a down cast. Strings have to be read as strings. Binary data needs to be accessed with the binary read/write operations. However, all data can be accessed through binary access.

Definition at line 214 of file plcBase.h.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 DataValue()

```
plc::DataValue::DataValue (
    data_type_enum rt,
    size_type len = 0 ) [inline], [explicit]
```

Constructor

Parameters

<i>rt</i>	Data type enumeration value
<i>len</i>	Length of data

Definition at line 226 of file plcBase.h.

References Init().

Here is the call graph for this function:



8.15.3 Member Function Documentation

8.15.3.1 GetValid()

```
bool plc::DataValue::GetValid (
    atomic_bool & dirty ) const [protected]
```

Get the valid flag and reset the dirty flag

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
--------------	---------------------------------------

Returns

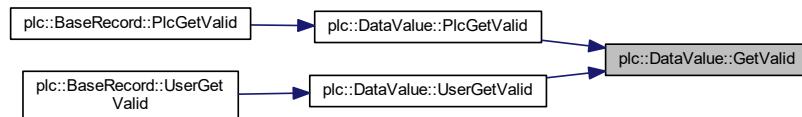
valid True for valid data, False for invalid

Definition at line 491 of file plcBase.cpp.

References `plc::DataValueTypeDef::memory_order`, and `myvalid`.

Referenced by `PlcGetValid()`, and `UserGetValid()`.

Here is the caller graph for this function:

**8.15.3.2 Init()**

```
void plc::DataValue::Init (
    data_type_enum rt,
    size_type len = 0 )
```

Initializes data value

Parameters

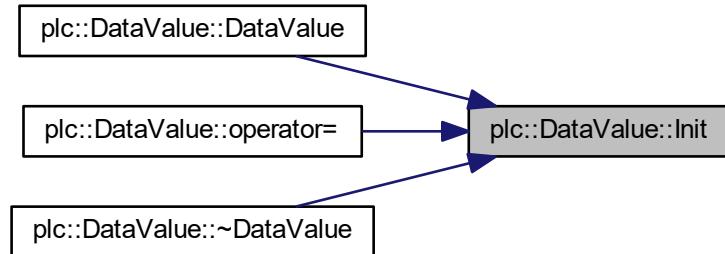
<i>rt</i>	Data type enumeration value
<i>len</i>	Length of data (use only for binary)

Definition at line 174 of file plcBase.cpp.

References `plc::dtBinary`, `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtInvalid`, `plc::dtString`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `plc::dtWString`, `mydata`, `myplcdirty`, `mysize`, `mytype`, and `myuserdirty`.

Referenced by `DataValue()`, `operator=()`, and `~DataValue()`.

Here is the caller graph for this function:



8.15.3.3 PlcGetValid()

```
bool plc::DataValue::PlcGetValid() const [inline]
```

Get the valid flag and reset the dirty flag

Returns

valid True for valid data, False for invalid

Definition at line 366 of file plcBase.h.

References GetValid(), and myuserdirty.

Referenced by plc::BaseRecord::PlcGetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.4 PlcRead() [1/4]

```
template<typename T >
bool plc::DataValue::PlcRead (
    T & data ) const [inline]
```

Read data by the plc

Parameters

<i>data</i>	Data value reference (return)
-------------	-------------------------------

Definition at line 311 of file plcBase.h.

References myplcdirty, and Read().

Referenced by plc::BaseRecord::PlcRead().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.5 PlcRead() [2/4]

```
template<size_type N>
bool plc::DataValue::PlcRead (
    type_string_value(&) data[N] ) const [inline]
```

Read fixed length character array data by the plc

Parameters

<i>data</i>	Data value reference for a fixed length character array (return)
-------------	--

Definition at line 315 of file plcBase.h.

References myplcdirty, and ReadBinary().

Here is the call graph for this function:



8.15.3.6 PlcRead() [3/4]

```
bool plc::DataValue::PlcRead (
    type_string_value * data,
    size_type max ) const [inline]
```

Read character array (pchar) by the plc

Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 320 of file plcBase.h.

References myplcdirty, and Read().

Here is the call graph for this function:



8.15.3.7 PlcRead() [4/4]

```
bool plc::DataValue::PlcRead (
    type_wstring_value * data,
    size_type max ) const [inline]
```

Read character array (pwchar) by the plc

Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 325 of file plcBase.h.

References myplcdirty, and Read().

Here is the call graph for this function:

**8.15.3.8 PlcReadBinary()**

```
size_type plc::DataValue::PlcReadBinary (
    type_binary p,
    size_type len ) const [inline]
```

Read data as binary by the plc

Parameters

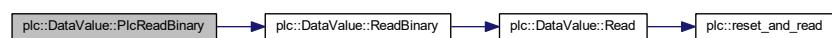
<i>p</i>	value pointer (destination buffer)
<i>len</i>	Length in bytes

Definition at line 349 of file plcBase.h.

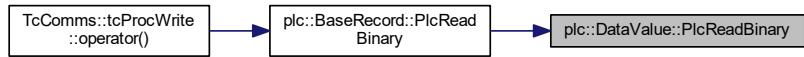
References myplcdirty, and ReadBinary().

Referenced by plc::BaseRecord::PlcReadBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.9 PlcSetValid()

```
void plc::DataValue::PlcSetValid ( bool valid ) [inline]
```

Set the valid flag and set the dirty flag when flag changes

Parameters

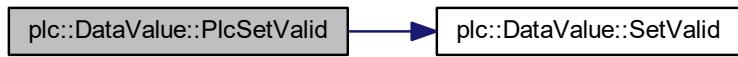
valid	True for valid data, False for invalid
-------	--

Definition at line 363 of file plcBase.h.

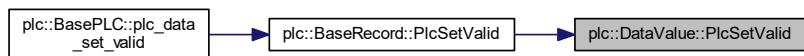
References myplcdirty, and SetValid().

Referenced by plc::BaseRecord::PlcSetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.10 PlcWrite() [1/4]

```
template<typename T >
bool plc::DataValue::PlcWrite (
    const T & data ) [inline]
```

Write data by the plc

Parameters

<i>data</i>	Data value reference
-------------	----------------------

Definition at line 329 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Referenced by plc::BaseRecord::PlcWrite().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.11 PlcWrite() [2/4]

```
template<size_type N>
bool plc::DataValue::PlcWrite (
    const type_string_value(&) data[N] ) [inline]
```

Write fixed length character array data by the plc

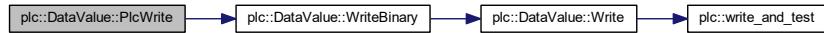
Parameters

<i>data</i>	Data value reference
-------------	----------------------

Definition at line 333 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Here is the call graph for this function:



8.15.3.12 PlcWrite() [3 / 4]

```
bool plc::DataValue::PlcWrite (
    const type_string_value * data,
    size_type max ) [inline]
```

Write character array (pchar) by the plc

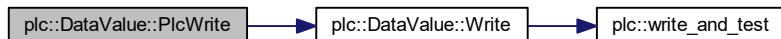
Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 338 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:



8.15.3.13 PlcWrite() [4 / 4]

```
bool plc::DataValue::PlcWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Write character array (wpchar) by the plc

Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 343 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:

**8.15.3.14 PlcWriteBinary()**

```
size_type plc::DataValue::PlcWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Write data as binary by the plc

Parameters

<i>p</i>	value pointer (source buffer)
<i>len</i>	Length in bytes

Definition at line 354 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Referenced by plc::BaseRecord::PlcWriteBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.15 Read() [1/5]

```
template<typename T >
bool plc::DataValue::Read (
    atomic_bool & dirty,
    T & data ) const [protected]
```

Read data

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Data value reference (return)

[DataValue::Read](#) (bool, Integral and floating point types)

Definition at line 87 of file plcBaseTemplate.h.

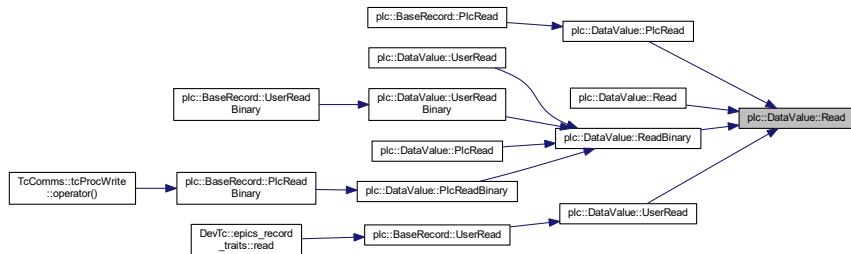
References `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `mydata`, `mytype`, and `plc::reset_and_read()`.

Referenced by `PlcRead()`, `Read()`, `ReadBinary()`, and `UserRead()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.16 Read() [2/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_string & data ) const [protected]
```

Read string (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Data value reference (return)

Definition at line 272 of file plcBase.cpp.

References plc::dtString, mydata, mytype, and plc::reset_and_read().

Here is the call graph for this function:



8.15.3.17 Read() [3/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_wstring & data ) const [protected]
```

Read wstring (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Data value reference (return)

Definition at line 284 of file plcBase.cpp.

References plc::dtString, plc::dtWString, mydata, mytype, and plc::reset_and_read().

Here is the call graph for this function:

**8.15.3.18 Read() [4/5]**

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_string_value * data,
    size_type max ) const [protected]
```

Read character array (pchar)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 299 of file plcBase.cpp.

References Read().

Here is the call graph for this function:



8.15.3.19 Read() [5/5]

```
bool plc::DataValue::Read (
    atomic_bool & dirty,
    type_wstring_value * data,
    size_type max ) const [protected]
```

Read character array (pwchar)

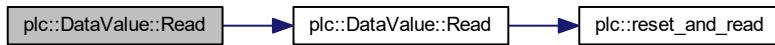
Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 313 of file plcBase.cpp.

References Read().

Here is the call graph for this function:



8.15.3.20 ReadBinary()

```
DataValue::size_type plc::DataValue::ReadBinary (
    atomic_bool & dirty,
    type_binary p,
    size_type len ) const [protected]
```

Read data as binary

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>p</i>	value pointer (destination buffer)
<i>len</i>	Length in bytes

Definition at line 383 of file plcBase.cpp.

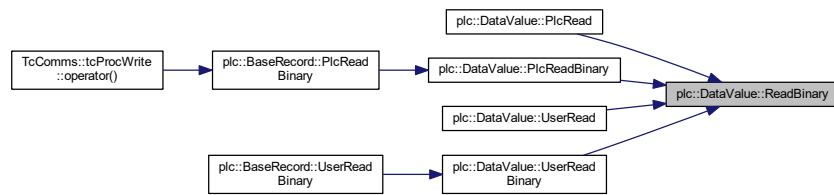
References plc::dtBinary, plc::dtBool, plc::dtDouble, plc::dtFloat, plc::dtInt16, plc::dtInt32, plc::dtInt64, plc::dtInt8, plc::dtInvalid, plc::dtString, plc::dtUInt16, plc::dtUInt32, plc::dtUInt64, plc::dtUInt8, plc::dtWString, plc::DataValue<TypeDef>::memory_order, mydata, mysize, mytype, and Read().

Referenced by PlcRead(), PlcReadBinary(), UserRead(), and UserReadBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.21 SetValid()

```

void plc::DataValue::SetValid (
    atomic_bool & dirty,
    bool valid ) [protected]
  
```

Set the valid flag and set the dirty flag when flag changes

Parameters

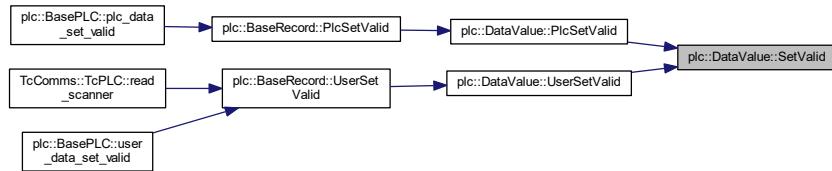
<i>dirty</i>	Reference to dirty flag (user or plc)
<i>valid</i>	True for valid data, False for invalid

Definition at line 479 of file plcBase.cpp.

References plc::DataValueTypeDef::memory_order, and myvalid.

Referenced by PlcSetValid(), and UserSetValid().

Here is the caller graph for this function:



8.15.3.22 UserGetValid()

```
bool plc::DataValue::UserGetValid( ) const [inline]
```

Get the valid flag and reset the dirty flag

Returns

valid True for valid data, False for invalid

Definition at line 307 of file plcBase.h.

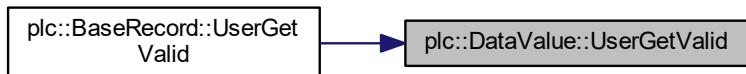
References GetValid(), and myplcdirty.

Referenced by plc::BaseRecord::UserGetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.23 UserRead() [1/4]

```
template<typename T >
bool plc::DataValue::UserRead (
    T & data ) const [inline]
```

Read data by the user

Parameters

<i>data</i>	Data value reference (return)
-------------	-------------------------------

Definition at line 251 of file plcBase.h.

References myuserdirty, and Read().

Referenced by plc::BaseRecord::UserRead().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.24 UserRead() [2/4]

```
template<size_type N>
bool plc::DataValue::UserRead (
    type_string_value(&) data[N] ) const [inline]
```

Read fixed length character array data by the user

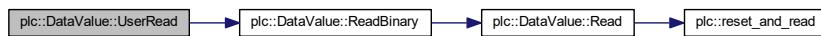
Parameters

<i>data</i>	Data value reference for a fixed length character array (return)
-------------	--

Definition at line 255 of file plcBase.h.

References myuserdirty, and ReadBinary().

Here is the call graph for this function:



8.15.3.25 UserRead() [3/4]

```
bool plc::DataValue::UserRead (
    type_string_value * data,
    size_type max ) const [inline]
```

Read character array (pchar) by the user

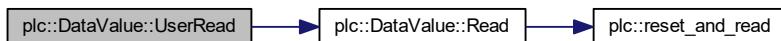
Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 260 of file plcBase.h.

References myuserdirty, and Read().

Here is the call graph for this function:



8.15.3.26 UserRead() [4/4]

```
bool plc::DataValue::UserRead (
    type_wstring_value * data,
    size_type max ) const [inline]
```

Read character array (pwchar) by the user

Parameters

<i>data</i>	Destination buffer
<i>max</i>	Maximum length

Definition at line 265 of file plcBase.h.

References myuserdirty, and Read().

Here is the call graph for this function:

**8.15.3.27 UserReadBinary()**

```
size_type plc::DataValue::UserReadBinary (
    type_binary p,
    size_type len ) const [inline]
```

Read data as binary by the user

Parameters

<i>p</i>	value pointer (destination buffer)
<i>len</i>	Length in bytes

Definition at line 290 of file plcBase.h.

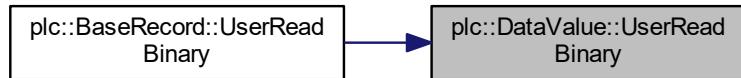
References myuserdirty, and ReadBinary().

Referenced by plc::BaseRecord::UserReadBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.28 UserSetValid()

```
void plc::DataValue::UserSetValid (
    bool valid ) [inline]
```

Set the valid flag and set the dirty flag when flag changes

Parameters

<code>valid</code>	True for valid data, False for invalid
--------------------	--

Definition at line 304 of file plcBase.h.

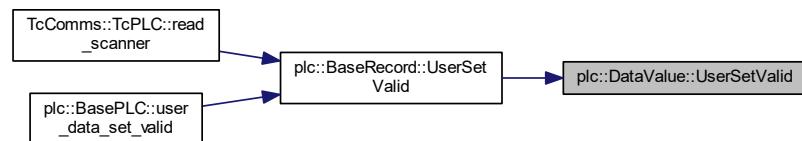
References myuserdirty, and SetValid().

Referenced by plc::BaseRecord::UserSetValid().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.29 UserWrite() [1/4]

```
template<typename T >
bool plc::DataValue::UserWrite (
    const T & data ) [inline]
```

Write data by the user

Parameters

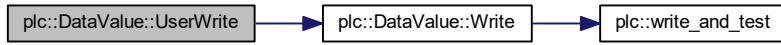
<i>data</i>	Data value reference
-------------	----------------------

Definition at line 269 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Referenced by plc::BaseRecord::UserWrite().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.30 UserWrite() [2/4]

```
template<size_type N>
bool plc::DataValue::UserWrite (
    const type_string_value(&) data[N] ) [inline]
```

Write fixed length character array data by the user

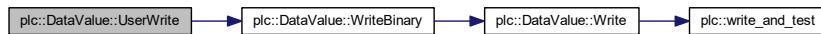
Parameters

<i>data</i>	Data value reference
-------------	----------------------

Definition at line 273 of file plcBase.h.

References myplcdirty, myuserdirty, and WriteBinary().

Here is the call graph for this function:



8.15.3.31 UserWrite() [3/4]

```
bool plc::DataValue::UserWrite (
    const type_string_value * data,
    size_type max ) [inline]
```

Write character array (pchar) by the user

Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 279 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:



8.15.3.32 UserWrite() [4/4]

```
bool plc::DataValue::UserWrite (
    const type_wstring_value * data,
    size_type max ) [inline]
```

Write character array (wpchar) by the user

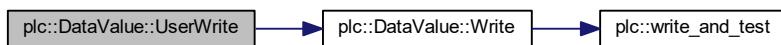
Parameters

<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 284 of file plcBase.h.

References myplcdirty, myuserdirty, and Write().

Here is the call graph for this function:

**8.15.3.33 UserWriteBinary()**

```
size_type plc::DataValue::UserWriteBinary (
    const type_binary p,
    size_type len ) [inline]
```

Write data as binary by the user

Parameters

<i>p</i>	value pointer (source buffer)
<i>len</i>	Length in bytes

Definition at line 295 of file plcBase.h.

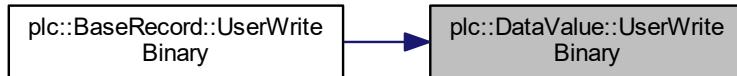
References myplcdirty, myuserdirty, and WriteBinary().

Referenced by plc::BaseRecord::UserWriteBinary().

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.34 Write() [1/5]

```
template<typename T >
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const T & data ) [protected]
```

Write data

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Data value reference (return)

[DataValue::UserWrite](#) (bool, Integral and floating point types)

Definition at line 120 of file `plcBaseTemplate.h`.

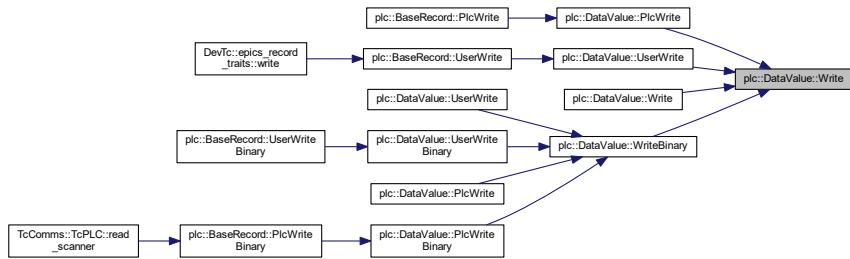
References `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `mydata`, `mytype`, `myvalid`, and `plc::write_and_test()`.

Referenced by `PlcWrite()`, `UserWrite()`, `Write()`, and `WriteBinary()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.3.35 Write() [2/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_string & data ) [protected]
```

Write string (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Data value reference

Definition at line 327 of file plcBase.cpp.

References plc::dtString, plc::dtWString, mydata, mytype, myvalid, and plc::write_and_test().

Here is the call graph for this function:



8.15.3.36 Write() [3/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_wstring & data ) [protected]
```

Write wstring (template specialization)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Data value reference

Definition at line 343 of file plcBase.cpp.

References plc::dtWString, mydata, mytype, myvalid, and plc::write_and_test().

Here is the call graph for this function:



8.15.3.37 Write() [4/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_string_value * data,
    size_type max ) [protected]
```

Write character array (pchar)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 356 of file plcBase.cpp.

References Write().

Here is the call graph for this function:



8.15.3.38 Write() [5/5]

```
bool plc::DataValue::Write (
    atomic_bool & dirty,
    const atomic_bool & pend,
    const type_wstring_value * data,
    size_type max ) [protected]
```

Write character array (pwchar)

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>data</i>	Source buffer
<i>max</i>	Maximum length

Definition at line 369 of file plcBase.cpp.

References Write().

Here is the call graph for this function:



8.15.3.39 WriteBinary()

```
DataValue::size_type plc::DataValue::WriteBinary (
    atomic_bool & dirty,
```

```
const atomic_bool & pend,
const type_binary p,
size_type len ) [protected]
```

Write data as binary

Parameters

<i>dirty</i>	Reference to dirty flag (user or plc)
<i>pend</i>	Reference to pending read flag (plc or user)
<i>p</i>	value pointer (source buffer)
<i>len</i>	Length in bytes

Definition at line 431 of file plcBase.cpp.

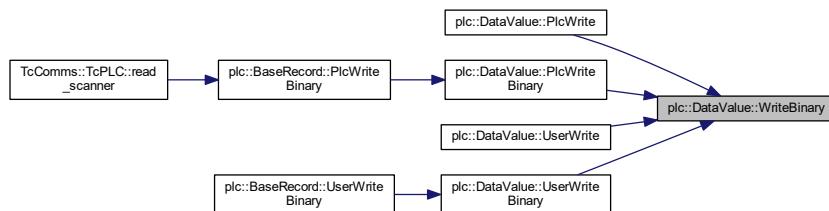
References `plc::dtBinary`, `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt64`, `plc::dtInt8`, `plc::dtInvalid`, `plc::dtString`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt64`, `plc::dtUInt8`, `plc::dtWString`, `plc::DataValue`←
`TypeDef::memory_order`, `mydata`, `mysize`, `mytype`, and `Write()`.

Referenced by `PlcWrite()`, `PlcWriteBinary()`, `UserWrite()`, and `UserWriteBinary()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.15.4 Member Data Documentation

8.15.4.1 mysize

`size_type` `plc::DataValue::mysize` [protected]

Size of allocated memory for simple types; size of string class for strings and size of data for binary

Definition at line 456 of file plcBase.h.

Referenced by `get_size()`, `Init()`, `IsValid()`, `operator=()`, `ReadBinary()`, and `WriteBinary()`.

The documentation for this class was generated from the following files:

- [plcBase.h](#)
- [plcBase.cpp](#)
- [plcBaseTemplate.h](#)

8.16 `plc::DataValueTraits< T >` Struct Template Reference

Data value traits.

```
#include <plcBase.h>
```

Public Types

- `typedef size_t size_type`
size type
- `typedef plc::data_type_enum data_type_enum`
enumerated type for data type
- `typedef T traits_type`
traits type
- `typedef std::atomic< T > traits_atomic`
atomic variable type

Public Member Functions

- `template<>`
`const DataValueTraits< DataValueTypeDef::type_bool >::data_type_enum data_enum`
Bool specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_int8 >::data_type_enum data_enum`
Int8 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_uint8 >::data_type_enum data_enum`
UInt8 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_int16 >::data_type_enum data_enum`
Int16 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_uint16 >::data_type_enum data_enum`
UInt16 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_int32 >::data_type_enum data_enum`
Int32 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_uint32 >::data_type_enum data_enum`
UInt32 specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_float >::data_type_enum data_enum`
Float specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_double >::data_type_enum data_enum`
Double specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_string >::data_type_enum data_enum`
String specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_wstring >::data_type_enum data_enum`
WString specialization for DataValueTraits.
- `template<>`
`const DataValueTraits< DataValueTypeDef::type_binary >::data_type_enum data_enum`
Binary specialization for DataValueTraits.

Static Public Attributes

- static const `data_type_enum data_enum = dtInvalid`
`data type enumertaion value`

8.16.1 Detailed Description

```
template<typename T>
struct plc::DataValueTraits< T >
```

Data value traits.

Traits class for data value

Definition at line 103 of file plcBase.h.

8.16.2 Member Data Documentation

8.16.2.1 data_enum

```
template<typename T>
const DataValueTraits< T >::data_type_enum plc::DataValueTraits< T >::data_enum = dtInvalid
[static]
```

data type enumertaion value

`DataValue::data_enum`

Definition at line 114 of file plcBase.h.

The documentation for this struct was generated from the following files:

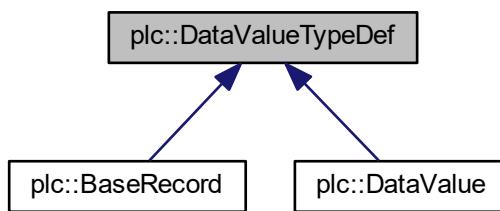
- [plcBase.h](#)
- [plcBaseTemplate.h](#)

8.17 plc::DataValueTypeDef Struct Reference

Collection of type definitions.

```
#include <plcBase.h>
```

Inheritance diagram for plc::DataValueTypeDef:



Public Types

- `typedef size_t size_type`
size type
- `typedef plc::data_type_enum data_type_enum`
enumerated type for data type
- `typedef bool type_bool`
bool type
- `typedef signed char type_int8`
1-byte integer type
- `typedef unsigned char type_uint8`
1-byte unsigned integer type
- `typedef short type_int16`
3-byte integer type
- `typedef unsigned short type_uint16`
3-byte unsigned integer type
- `typedef int type_int32`
4-byte integer type
- `typedef unsigned int type_uint32`
4-byte unsigned integer type
- `typedef long long type_int64`
8-byte integer type
- `typedef unsigned long long type_uint64`
8-byte unsigned integer type
- `typedef float type_float`
4-byte single precision floating point type
- `typedef double type_double`
4-byte double precision floating point type
- `typedef std::string type_string`
string type
- `typedef std::wstring type_wstring`
wstring type
- `typedef void * type_binary`
binary type
- `typedef std::string::value_type type_string_value`
string character type
- `typedef std::wstring::value_type type_wstring_value`
wstring character type
- `typedef DataValueTraits< type_bool >::traits_atomic atomic_bool`
atomic bool type
- `typedef DataValueTraits< type_int8 >::traits_atomic atomic_int8`
atomic 1-byte integer type
- `typedef DataValueTraits< type_uint8 >::traits_atomic atomic_uint8`
atomic 1-byte unsigned integer type
- `typedef DataValueTraits< type_int16 >::traits_atomic atomic_int16`
atomic 2-byte integer type
- `typedef DataValueTraits< type_uint16 >::traits_atomic atomic_uint16`
atomic 2-byte unsigned integer type
- `typedef DataValueTraits< type_int32 >::traits_atomic atomic_int32`
atomic 4-byte integer type
- `typedef DataValueTraits< type_uint32 >::traits_atomic atomic_uint32`

- `atomic 4-byte unsigned integer type`
- `typedef DataValueTraits< type_int64 >::traits_atomic atomic_int64`
`atomic 8-byte integer type`
- `typedef DataValueTraits< type_uint64 >::traits_atomic atomic_uint64`
`atomic 8-byte unsigned integer type`
- `typedef DataValueTraits< type_float >::traits_atomic atomic_float`
`atomic 4-byte single precision floating point type`
- `typedef DataValueTraits< type_double >::traits_atomic atomic_double`
`atomic 8-byte double precision floating point type`
- `typedef DataValueTraits< type_string >::traits_atomic atomic_string`
`atomic string type`
- `typedef DataValueTraits< type_wstring >::traits_atomic atomic_wstring`
`atomic wstring type`
- `typedef DataValueTraits< type_binary >::traits_atomic atomic_binary`
`atomic binary type`
- `typedef DataValueTypeDef::type_uint64 time_type`
`Define timestamp type.`

Static Public Attributes

- `static const std::memory_order memory_order = std::memory_order_seq_cst`
`memory order used for atomic access`

8.17.1 Detailed Description

Collection of type definitions.

Type definitions for data value

Definition at line 120 of file plcBase.h.

The documentation for this struct was generated from the following file:

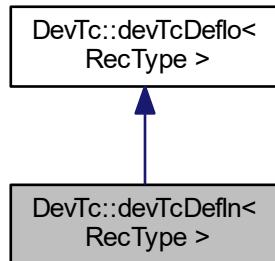
- `plcBase.h`

8.18 DevTc::devTcDefIn< RecType > Struct Template Reference

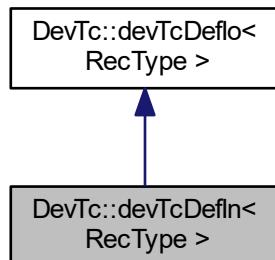
Device support input record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefIn< RecType >:



Collaboration diagram for DevTc::devTcDefIn< RecType >:



Public Member Functions

- [devTcDefIn \(\)](#)

Constructor.

Static Public Member Functions

- static long [init_read_record \(rec_type_ptr prec\)](#)

init callback for read records

- static long [read \(rec_type_ptr precord\)](#)

read callback

Additional Inherited Members

8.18.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefIn< RecType >
```

Device support input record.

Device Support Record for TwinCAT/ADS input This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 333 of file devTc.h.

The documentation for this struct was generated from the following file:

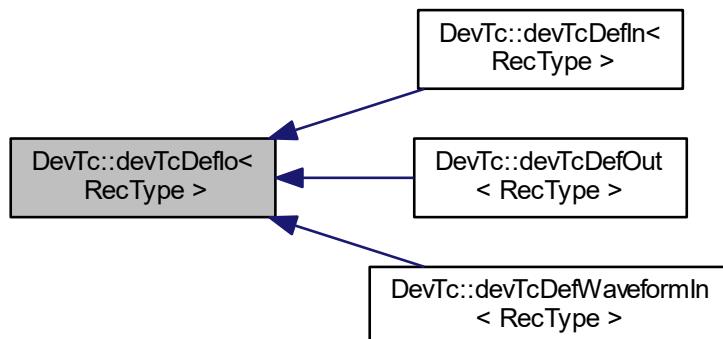
- [devTc.h](#)

8.19 DevTc::devTcDeflo< RecType > Struct Template Reference

Device support record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDeflo< RecType >:



Public Types

- **typedef epics_record_traits< RecType >::traits_type rec_type**
Record type: aiRecord, etc.
- **typedef rec_type * rec_type_ptr**
Pointer to record type.

Public Attributes

- long [number](#)
Number of support functions.
- DEVSUPFUN [report_fn](#)
Report support function.
- DEVSUPFUN [init_fn](#)
Init support function.
- DEVSUPFUN [init_record_fn](#)
Record init support function.
- DEVSUPFUN [get_ioint_info_fn](#)
IO/INT support function.
- DEVSUPFUN [io_fn](#)
Read/write support function.
- DEVSUPFUN [special_linconv_fn](#)
Linear conversion support function.

Protected Member Functions

- [devTcDeflo \(\)](#)
Hide constructor.

Static Protected Member Functions

- static long [get_ioint_info](#) (int cmd, dbCommon *prec, IOSCANPVT *ppvt)
IO/INT info callback.

8.19.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDeflo< RecType >
```

Device support record.

Deviced Support Record for generic TwinCAT/ADS IO This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 298 of file devTc.h.

The documentation for this struct was generated from the following file:

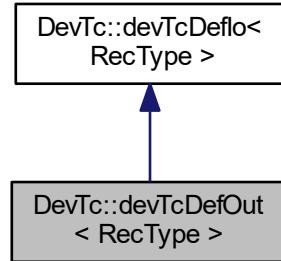
- [devTc.h](#)

8.20 DevTc::devTcDefOut< RecType > Struct Template Reference

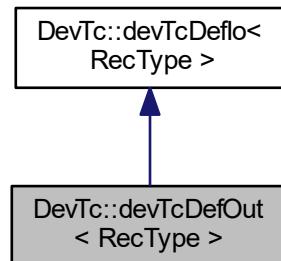
device support output record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefOut< RecType >:



Collaboration diagram for DevTc::devTcDefOut< RecType >:



Public Member Functions

- [devTcDefOut \(\)](#)

Constructor.

Static Public Member Functions

- static long [init_write_record \(rec_type_ptr prec\)](#)
init callback for write records
- static long [write \(rec_type_ptr precord\)](#)
write callback

Additional Inherited Members

8.20.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefOut< RecType >
```

device support output record.

Device Support Record for TwinCAT/ADS output This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 349 of file devTc.h.

The documentation for this struct was generated from the following file:

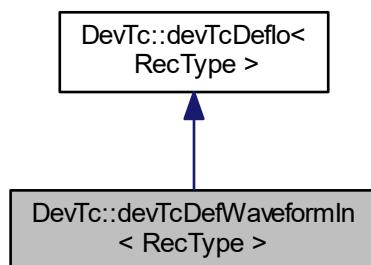
- [devTc.h](#)

8.21 DevTc::devTcDefWaveformIn< RecType > Struct Template Reference

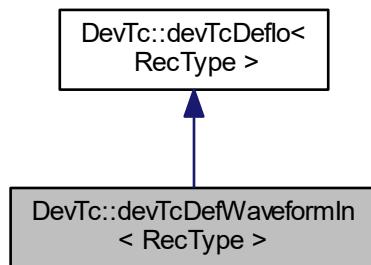
device support waveform record.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::devTcDefWaveformIn< RecType >:



Collaboration diagram for DevTc::devTcDefWaveformIn< RecType >:



Public Member Functions

- `devTcDefWaveformIn ()`

Constructor.

Static Public Member Functions

- static long `init_read_waveform_record (rec_type_ptr prec)`
init callback for read records
- static long `read_waveform (rec_type_ptr precord)`
read callback

Additional Inherited Members

8.21.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::devTcDefWaveformIn< RecType >
```

device support waveform record.

Device Support Record for TwinCAT/ADS waveform input This structure defines the callback functions for the TC device support. This is a base class for both read and write records.

Definition at line 365 of file devTc.h.

The documentation for this struct was generated from the following file:

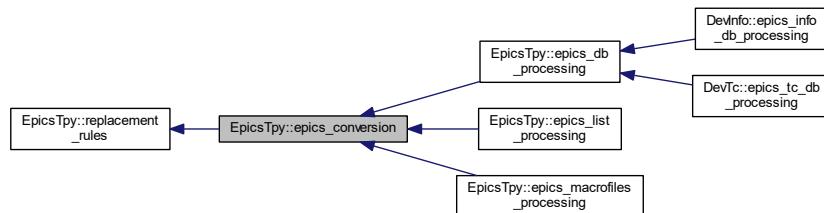
- `devTc.h`

8.22 EpicsTpy::epics_conversion Class Reference

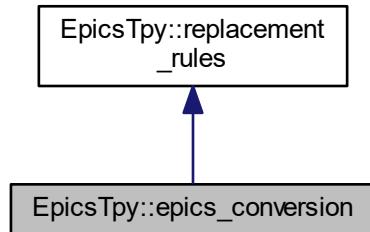
Epics conversion.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_conversion:



Collaboration diagram for EpicsTpy::epics_conversion:



Public Member Functions

- `epics_conversion ()`
Default constructor.
- `epics_conversion (case_type caseconv, bool noindex)`
- `epics_conversion (tc_epics_conv epics_conv, case_type caseconv, bool noldot, bool noindex)`
- `epics_conversion (int argc, const char *const argv[], bool argp[] = 0)`
- `int getopt (int argc, const char *const argv[], bool argp[] = 0)`
- `tc_epics_conv get_conversion_rule () const`
Get the conversion rule.
- `void set_conversion_rule (tc_epics_conv epics_conv)`
Set the conversion rule.
- `case_type get_case_rule () const`
Get the conversion rule.
- `void set_case_rule (case_type epics_conv)`
Set the conversion rule.
- `bool get_dot_rule () const`
Get the leadin dot rule.
- `void set_dot_rule (bool noldot)`
Set the leading dot rule.
- `bool get_array_rule () const`
Get the array index rule.
- `void set_array_rule (bool noindex)`
Set the array conversion rule.
- `std::string to_epics (const std::stringcase &name) const`

Protected Attributes

- `tc_epics_conv conv_rule`
Conversion rule.
- `case_type case_epics_names`
Case conversion rule.
- `bool no_leading_dot`
Leading dot conversion rule.
- `bool no_array_index`
Array index conversion rule.

Additional Inherited Members

8.22.1 Detailed Description

Epics conversion.

Epics channel conversion arguments Epics channels are generated from opc through a conversion rule

Definition at line 101 of file TpyToEpics.h.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 epics_conversion() [1/3]

```
EpicsTpy::epics_conversion::epics_conversion (
    case_type caseconv,
    bool noindex ) [inline]
```

Constructor

Parameters

<i>caseconv</i>	Case conversion specification
<i>noindex</i>	Eliminate array indices '[n]' with '_n'

Definition at line 110 of file TpyToEpics.h.

8.22.2.2 epics_conversion() [2/3]

```
EpicsTpy::epics_conversion::epics_conversion (
    tc_epics_conv epics_conv,
    case_type caseconv,
    bool noldot,
    bool noindex ) [inline]
```

Constructor

Parameters

<i>epics_conv</i>	Epics conversion rule
<i>caseconv</i>	Case conversion specification
<i>noldot</i>	Eliminate leading dot in a name
<i>noindex</i>	Eliminate array indices '[n]' with '_n'

Definition at line 118 of file TpyToEpics.h.

8.22.2.3 epics_conversion() [3/3]

```
EpicsTpy::epics_conversion::epics_conversion (
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 129 of file TpyToEpics.h.

References getopt().

Here is the call graph for this function:



8.22.3 Member Function Documentation

8.22.3.1 getopt()

```
int EpicsTpy::epics_conversion::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/rn: Does not apply any special conversion rules /rd: Replaces dots with underscores in channel names /rl: LIGO standard conversion rule (default) /cp: Preserve case in EPICS channel names /cu: Force upper case in EPICS

channel names (default) /cl: Force lower case in EPICS channel names /nd: Eliminates leading dot in channel name (default) /yd: Leaves leading dot in channel name /ni: Replaces array brackets with underscore (default) /yi: Leave array indices as is

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

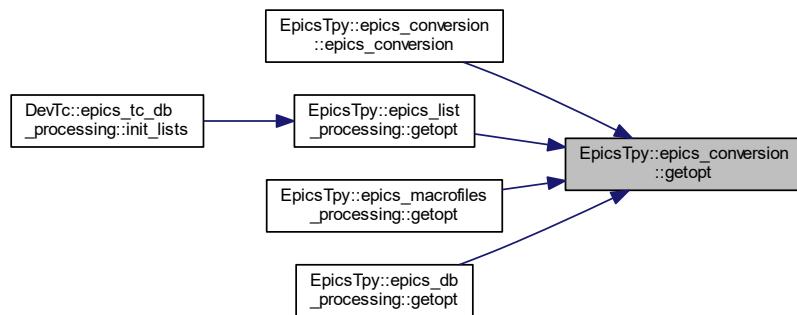
Number of arguments processed

Definition at line 65 of file TpyToEpics.cpp.

References `EpicsTpy::ligo_std`, `EpicsTpy::ligo_vac`, `EpicsTpy::lower_case`, `EpicsTpy::no_conversion`, `EpicsTpy::no_dot`, `EpicsTpy::preserve_case`, and `EpicsTpy::upper_case`.

Referenced by `epics_conversion()`, `EpicsTpy::epics_list_processing::getopt()`, `EpicsTpy::epics_macrofiles_processing::getopt()`, and `EpicsTpy::epics_db_processing::getopt()`.

Here is the caller graph for this function:

**8.22.3.2 to_epics()**

```
string EpicsTpy::epics_conversion::to_epics (
    const std::string& name ) const
```

Converts a TwinCAT or OPC name to an EPICS channel name

Parameters

<i>name</i>	TwinCAT/opc name
-------------	------------------

Returns

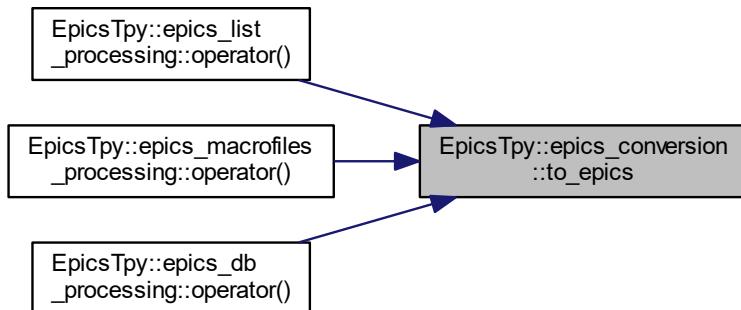
EPICS name

Definition at line 141 of file TpyToEpics.cpp.

References `EpicsTpy::ligo_std`, `EpicsTpy::ligo_vac`, `EpicsTpy::no_conversion`, `EpicsTpy::no_dot`, `EpicsTpy::preserve_case`, and `EpicsTpy::upper_case`.

Referenced by `EpicsTpy::epics_list_processing::operator()`, `EpicsTpy::epics_macrofiles_processing::operator()`, and `EpicsTpy::epics_db_processing::operator()`.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

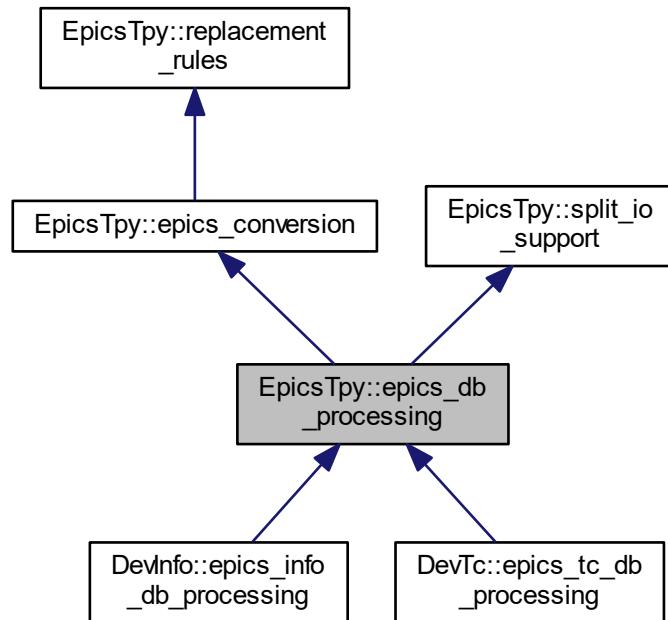
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.23 EpicsTpy::epics_db_processing Class Reference

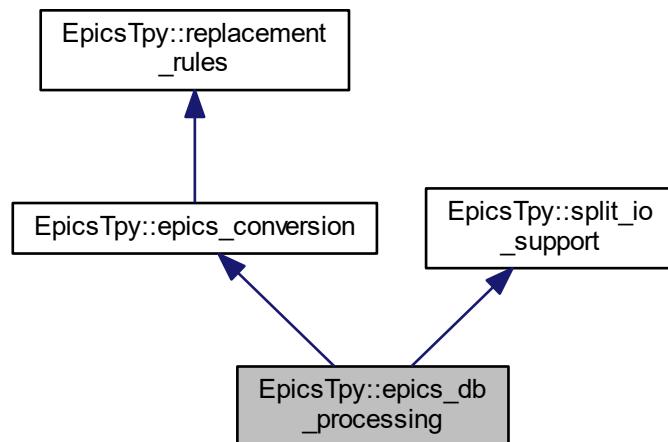
Epics database record processing

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_db_processing:



Collaboration diagram for EpicsTpy::epics_db_processing:



Public Member Functions

- [epics_db_processing \(\)](#)

Default constructor.

- `epics_db_processing (const std::stringcase &fname, int argc, const char *const argv[], bool argp[] = 0)`
 - `int getopt (int argc, const char *const argv[], bool argp[] = 0)`
 - `int mygetopt (int argc, const char *const argv[], bool argp[] = 0)`
 - `device_support_type get_device_support () const`
- Get device support conversion rule.*
- `void set_device_support (device_support_type devsup)`
- Set device support conversion rule.*
- `bool operator() (const ParseUtil::process_arg &arg)`

Protected Member Functions

- `bool process_field_string (std::stringcase name, std::stringcase val)`
- `bool process_field_numeric (std::stringcase name, int val)`
- `bool process_field_numeric (std::stringcase name, double val)`
- `bool process_field_numeric (std::stringcase name, std::stringcase val)`
- `bool process_field_alarm (std::stringcase name, std::stringcase severity)`

Protected Attributes

- `device_support_type device_support`
- Device support field conversion rule.*

Additional Inherited Members

8.23.1 Detailed Description

Epics database record processing

Class for generating an EPICS database record

Definition at line 742 of file TpyToEpics.h.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 epics_db_processing()

```
EpicsTpy::epics_db_processing::epics_db_processing (
    const std::stringcase & fname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Constructor Command line arguments will override default parameters when specified. The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored. Processed options with `epics_conversion::getopt`, `split_io_support::getopt` and `mygetopt()`.

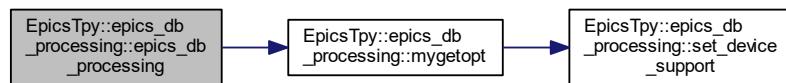
Parameters

<i>fname</i>	Output filename
<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 1220 of file TpyToEpics.cpp.

References [my getopt\(\)](#).

Here is the call graph for this function:



8.23.3 Member Function Documentation

8.23.3.1 getopt()

```
int EpicsTpy::epics_db_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line Processed options with [epics_conversion::getopt](#) and my getopt.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

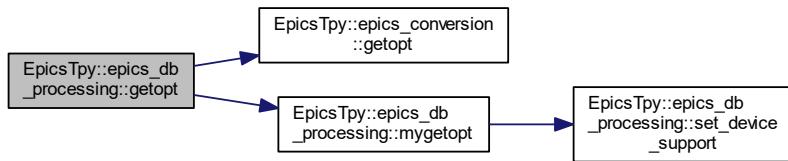
Returns

Number of arguments processed

Definition at line 1232 of file TpyToEpics.cpp.

References [EpicsTpy::epics_conversion::getopt\(\)](#), and [my getopt\(\)](#).

Here is the call graph for this function:



8.23.3.2 mygetopt()

```
int EpicsTpy::epics_db_processing::mygetopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/devopc: Uses OPC name in INPUT/OUTPUT field (default) /devtc: Uses TwinCAT name in INPUT/OUTPUT fields instead of OPC

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

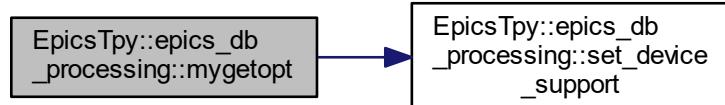
Number of arguments processed

Definition at line 1242 of file TpyToEpics.cpp.

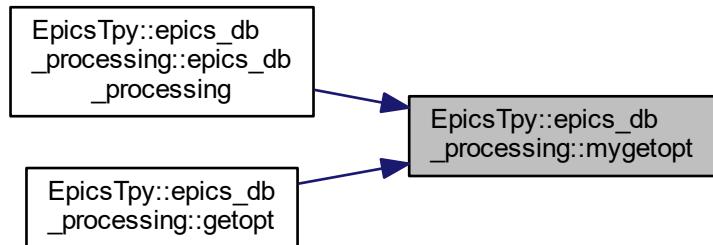
References EpicsTpy::device_support_opc_name, EpicsTpy::device_support_tc_name, and set_device_support().

Referenced by epics_db_processing(), and getopt().

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.3.3 operator()()

```
bool EpicsTpy::epics_db_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

Parameters

<code>arg</code>	Process argument describign the variable and type
------------------	---

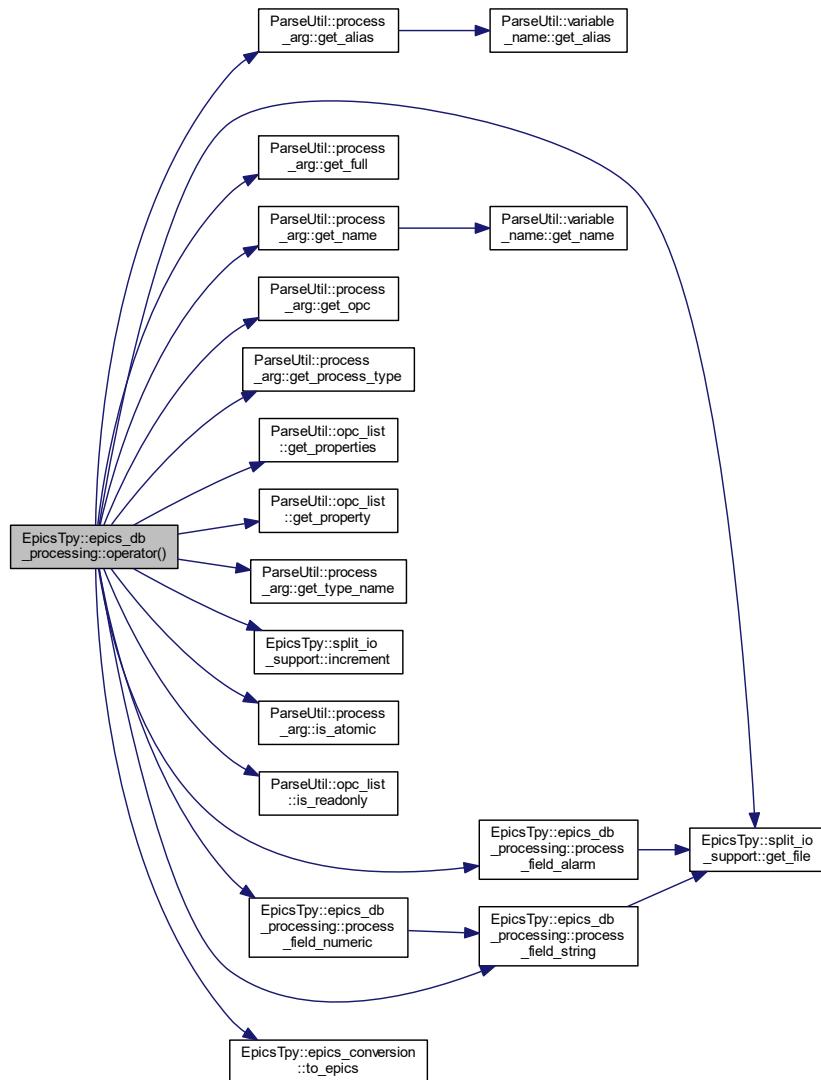
Returns

True if successfully processed

Definition at line 1275 of file TpyToEpics.cpp.

References device_support, EpicsTpy::device_support_opc_name, EpicsTpy::device_support_tc_name, EpicsTpy::EPICS_DB_COSV, EpicsTpy::EPICS_DB_DESC, EpicsTpy::EPICS_DB_DRVH, EpicsTpy::EPICS_DB_DRVL, EpicsTpy::EPICS_DB_DTYP, EpicsTpy::EPICS_DB_EGU, EpicsTpy::EPICS_DB_HHSV, EpicsTpy::EPI_CS_DB_HIGH, EpicsTpy::EPICS_DB_HIHI, EpicsTpy::EPICS_DB_HOPR, EpicsTpy::EPICS_DB_HSV, EpicsTpy::EPICS_DB_HYST, EpicsTpy::EPICS_DB_INP, EpicsTpy::EPICS_DB_LLSV, EpicsTpy::EPICS_DB_LOLO, EpicsTpy::EPICS_DB_LOPR, EpicsTpy::EPICS_DB_LOW, EpicsTpy::EPICS_DB_LSV, EpicsTpy::EPICS_DB_MAJOR, EpicsTpy::EPICS_DB_MINOR, EpicsTpy::EPICS_DB_ONAM, EpicsTpy::EPICS_DB_OSV, EpicsTpy::EPICS_DB_OUT, EpicsTpy::EPICS_DB_PINI, EpicsTpy::EPICS_DB_PREC, EpicsTpy::EPICS_DB_SCAN, EpicsTpy::EPICS_DB_TSE, EpicsTpy::EPICS_DB_UNSV, EpicsTpy::EPICS_DB_ZNAM, EpicsTpy::EPICS_DB_ZRST, EpicsTpy::EPICS_DB_ZRSV, EpicsTpy::EPICS_DB_ZRVL, EpicsTpy::EPICS_DB_ZSV, ParseUtil::process_arg::get_alias(), EpicsTpy::split_io_support::get_file(), ParseUtil::process_arg::get_full(), ParseUtil::process_arg::get_name(), ParseUtil::process_arg::get_opc(), ParseUtil::process_arg::get_process_type(), ParseUtil::opc_list::get_properties(), ParseUtil::opc_list::get_property(), ParseUtil::process_arg::get_type_name(), EpicsTpy::split_io_support::increment(), ParseUtil::process_arg::is_atomic(), ParseUtil::opc_list::is_READONLY(), EpicsTpy::MAX_EPICS_CHANNEL, EpicsTpy::MAX_EPICS_DESC, ParseUtil::OPC_PROP_ALIAS, ParseUtil::OPC_PROP_ALMC_OSV, ParseUtil::OPC_PROP_ALMDB, ParseUtil::OPC_PROP_ALMFFSV, ParseUtil::OPC_PROP_ALMH, ParseUtil::OPC_PROP_ALMH, ParseUtil::OPC_PROP_ALMHHSV, ParseUtil::OPC_PROP_ALMHHSV, ParseUtil::OPC_PROP_ALML, ParseUtil::OPC_PROP_ALMLL, ParseUtil::OPC_PROP_ALMLLSV, ParseUtil::OPC_PROP_ALMLSV, ParseUtil::OPC_PROP_ALMOSV, ParseUtil::OPC_PROP_ALMUNSV, ParseUtil::OPC_PROP_ALMZRSV, ParseUtil::OPC_PROP_ALMZSV, ParseUtil::OPC_PROP_CLOSE, ParseUtil::OPC_PROP_DESC, ParseUtil::OPC_PROP_DTYP, ParseUtil::OPC_PROP_FFST, ParseUtil::OPC_PROP_HIEU, ParseUtil::OPC_PROP_HIRANGE, ParseUtil::OPC_PROP_INOUT, ParseUtil::OPC_PROP_LOEU, ParseUtil::OPC_PROP_LORANGE, ParseUtil::OPC_PROP_OPEN, ParseUtil::OPC_PROP_PINI, ParseUtil::OPC_PROP_PLNAME, ParseUtil::OPC_PROP_PREC, ParseUtil::OPC_PROP_RECTYPE, ParseUtil::OPC_PROP_SERVER, ParseUtil::OPC_PROP_TSE, ParseUtil::OPC_PROP_UNIT, ParseUtil::OPC_PROP_ZRST, process_field_alarm(), process_field_numeric(), process_field_string(), ParseUtil::pt_bool, ParseUtil::pt_enum, ParseUtil::pt_int, ParseUtil::pt_real, ParseUtil::pt_string, and EpicsTpy::epics_conversion::to_epics().

Here is the call graph for this function:



8.23.3.4 process_field_alarm()

```
bool EpicsTpy::epics_db_processing::process_field_alarm (
    std::stringcase name,
    std::stringcase severity ) [protected]
```

Process a record field of type alarm

Parameters

<i>name</i>	Name of field
<i>severity</i>	Severity of alarm

Returns

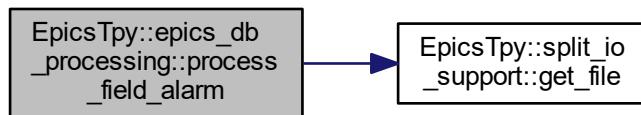
True if successful

Definition at line 1599 of file TpyToEpics.cpp.

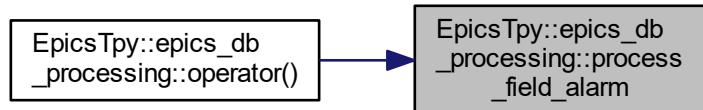
References `EpicsTpy::EPICS_DB_MAJOR`, `EpicsTpy::EPICS_DB_MINOR`, `EpicsTpy::EPICS_DB_NOALARM`, and `EpicsTpy::split_io_support::get_file()`.

Referenced by `operator()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.3.5 process_field_numeric() [1/3]

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    int val )  [protected]
```

Process a record field of numeric type

Parameters

<code>name</code>	Name of field
<code>val</code>	Value of field

Returns

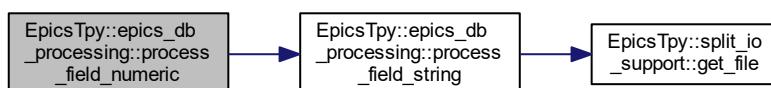
True if successful

Definition at line 1563 of file TpyToEpics.cpp.

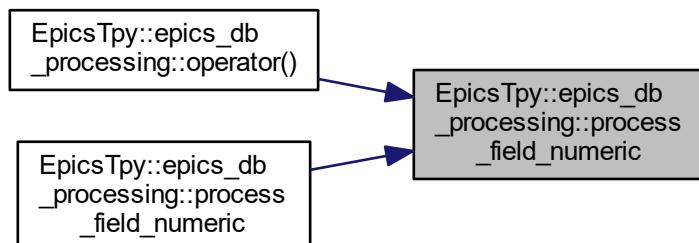
References process_field_string().

Referenced by operator()(), and process_field_numeric().

Here is the call graph for this function:



Here is the caller graph for this function:



8.23.3.6 process_field_numeric() [2/3]

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    double val )  [protected]
```

Process a record field of numeric type

Parameters

<i>name</i>	Name of field
<i>val</i>	Value of field

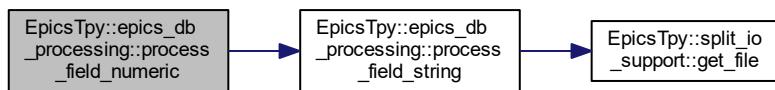
Returns

True if successful

Definition at line 1573 of file TpyToEpics.cpp.

References process_field_string().

Here is the call graph for this function:

**8.23.3.7 process_field_numeric() [3/3]**

```
bool EpicsTpy::epics_db_processing::process_field_numeric (
    std::stringcase name,
    std::stringcase val ) [protected]
```

Process a record field of type string

Parameters

<i>name</i>	Name of field
<i>val</i>	Value of field

Returns

True if successful

Definition at line 1583 of file TpyToEpics.cpp.

References process_field_numeric().

Here is the call graph for this function:



8.23.3.8 process_field_string()

```
bool EpicsTpy::epics_db_processing::process_field_string (
    std::stringcase name,
    std::stringcase val ) [protected]
```

Process a record field of type string

Parameters

<i>name</i>	Name of field
<i>val</i>	Value of field

Returns

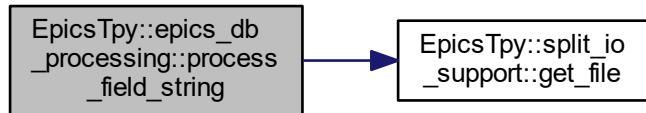
True if successful

Definition at line 1553 of file TpyToEpics.cpp.

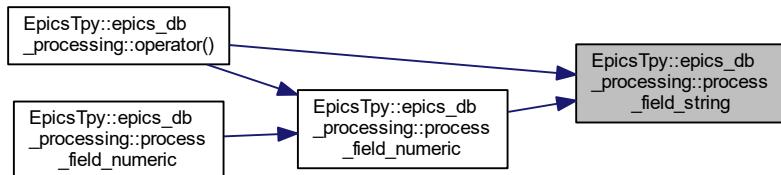
References EpicsTpy::split_io_support::get_file().

Referenced by operator()(), and process_field_numeric().

Here is the call graph for this function:



Here is the caller graph for this function:



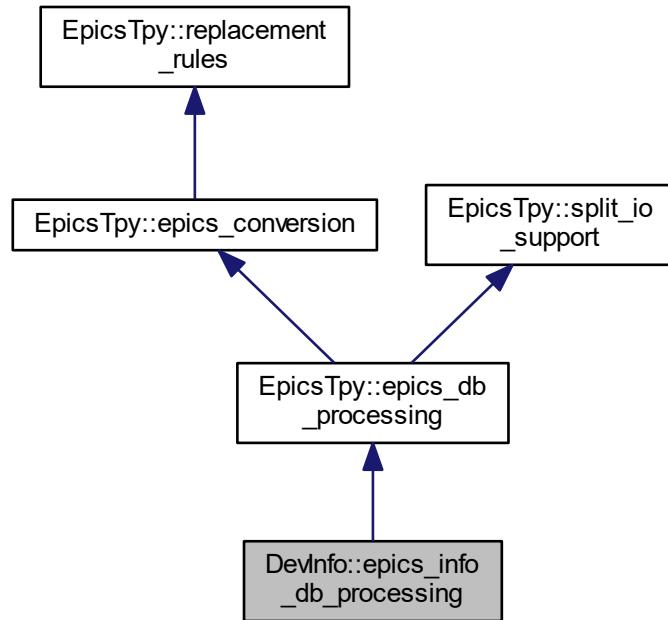
The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

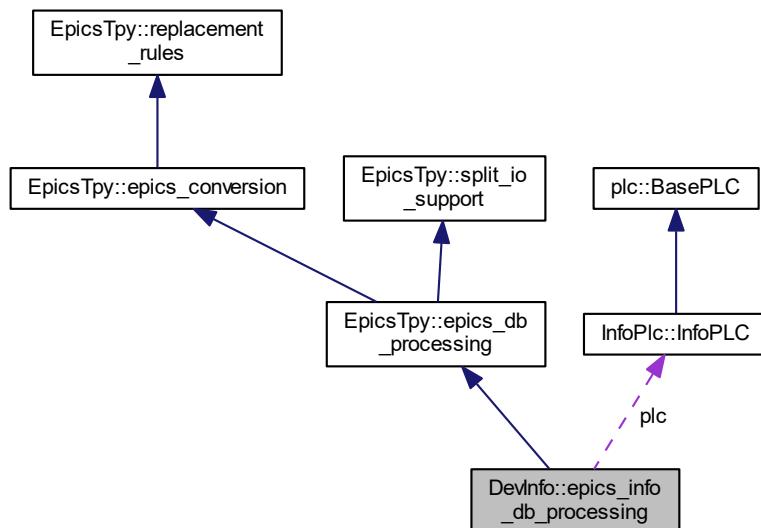
8.24 DevInfo::epics_info_db_processing Class Reference

EPICS/Info db processing.

Inheritance diagram for DevInfo::epics_info_db_processing:



Collaboration diagram for DevInfo::epics_info_db_processing:



Public Member Functions

- `epics_info_db_processing (InfoPLC &p)`
Default constructor.
- `bool operator() (const ParseUtil::process_arg &arg)`
- `int get_invalid_records () const`
Get number of EPICS records without info records.

Protected Attributes

- `InfoPLC * plc`
Pointer to PLC class.
- `int invnum`
Number of EPICS records without info records.

Additional Inherited Members

8.24.1 Detailed Description

EPICS/Info db processing.

Class for generating an EPICS database and info record

Definition at line 72 of file drvInfo.cpp.

8.24.2 Member Function Documentation

8.24.2.1 operator()()

```
bool DevInfo::epics_info_db_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

Parameters

<code>arg</code>	Process argument describign the variable and type
------------------	---

Returns

True if successful

Determine data type of this record

Make new record object

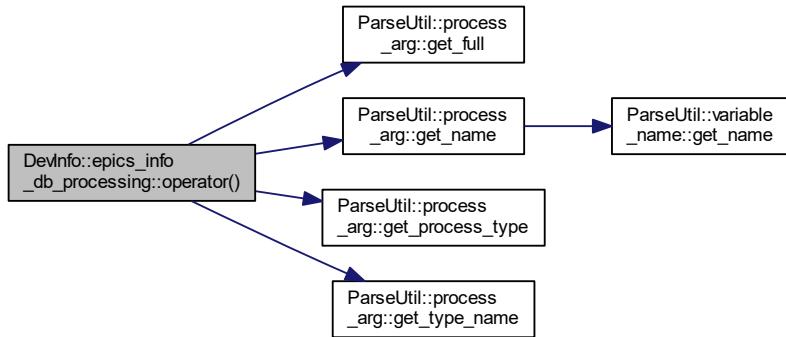
Make info interface

Tell record about info interface

Definition at line 97 of file drvInfo.cpp.

References `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt8`, `plc::dtInvalid`, `plc::dtString`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt8`, `ParseUtil::process_arg::get_full()`, `ParseUtil::process_arg::get_name()`, `ParseUtil::process_arg::get_process_type()`, `ParseUtil::process_arg::get_type_name()`, and `ParseUtil::pt_enum`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

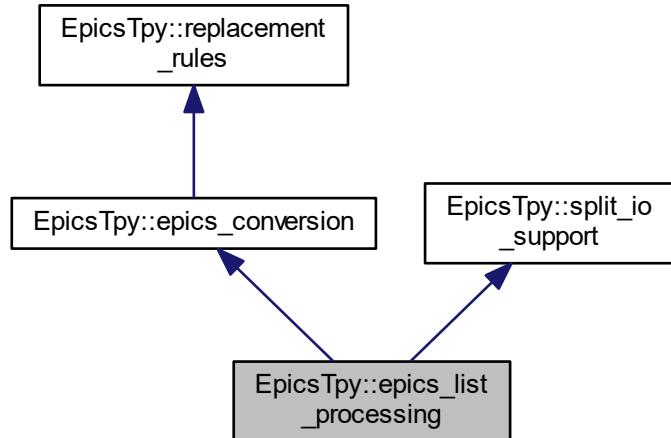
- [drvInfo.cpp](#)

8.25 EpicsTpy::epics_list_processing Class Reference

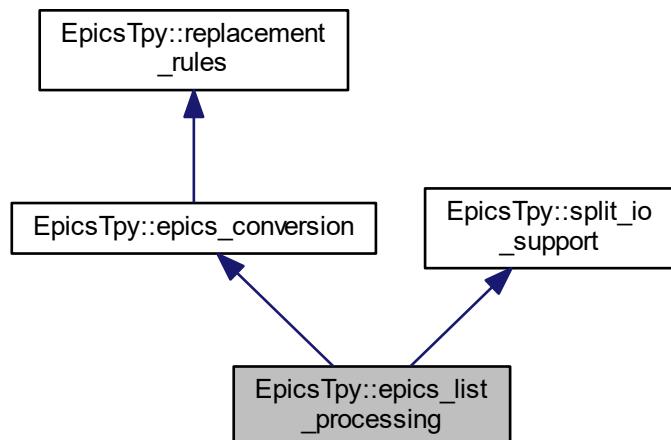
List processing.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_list_processing:



Collaboration diagram for EpicsTpy::epics_list_processing:



Public Member Functions

- [epics_list_processing \(\)](#)
Default constructor.
- [epics_list_processing \(listing_type ltype, bool ll=false\)](#)
- [epics_list_processing \(const std::string& fname, int argc, const char *const argv\[\], bool argp\[\] = 0\)](#)
- [int getopt \(int argc, const char *const argv\[\], bool argp\[\] = 0\)](#)

- int [mygetopt](#) (int argc, const char *const argv[], bool argp[] = 0)
- bool [operator\(\)](#) (const ParseUtil::process_arg &arg)
- [listing_type get_listing \(\) const](#)
Get listing type.
- void [set_listing \(listing_type lt\)](#)
Set listing.
- bool [is_verbose \(\) const](#)
Is long listing?
- void [set_verbose \(bool vrbs\)](#)
Set long listing.

Protected Attributes

- [listing_type listing](#)
Listing type.
- bool [verbose](#)
long listing

Additional Inherited Members

8.25.1 Detailed Description

List processing.

Class for generating a channel list

Definition at line 475 of file TpyToEpics.h.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 epics_list_processing() [1/2]

```
EpicsTpy::epics_list_processing::epics_list_processing (
    listing_type ltype,
    bool ll = false )  [inline], [explicit]
```

Constructor

Parameters

<i>ltype</i>	Type of listing
<i>ll</i>	long listing

Definition at line 484 of file TpyToEpics.h.

8.25.2.2 `epics_list_processing()` [2/2]

```
EpicsTpy::epics_list_processing::epics_list_processing (
    const std::string& fname,
    int argc,
    const char *const argv[],
    bool argp[] = {0})
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored Processed options with [epics_conversion::getopt](#), [split_io_support::getopt](#) and [my getopt\(\)](#).

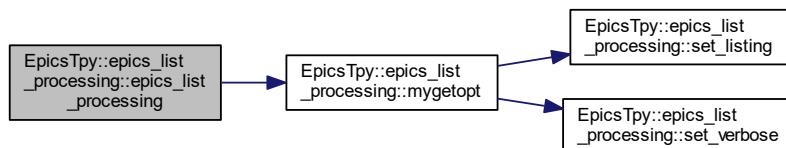
Parameters

<code>fname</code>	Output filename
<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in main()
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 565 of file TpyToEpics.cpp.

References [my getopt\(\)](#).

Here is the call graph for this function:



8.25.3 Member Function Documentation

8.25.3.1 `getopt()`

```
int EpicsTpy::epics_list_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = {0})
```

Parse a command line Processed options with [epics_conversion::getopt](#) and [my getopt\(\)](#).

Parameters

<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in main()
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Returns

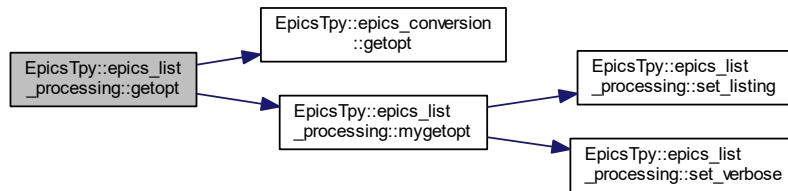
Number of arguments processed

Definition at line 577 of file TpyToEpics.cpp.

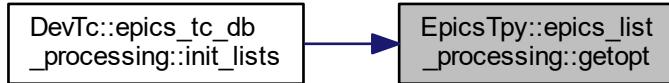
References EpicsTpy::epics_conversion::getopt(), and my getopt().

Referenced by DevTc::epics_tc_db_processing::init_lists().

Here is the call graph for this function:



Here is the caller graph for this function:

**8.25.3.2 my getopt()**

```

int EpicsTpy::epics_list_processing::mygetopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
  
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/I: Generates a standard listing (default) /II: Generates a long listing /lb: Generates an autoburn save/restore file /li: Generates a LIGO DAQ ini file

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in main()
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Returns

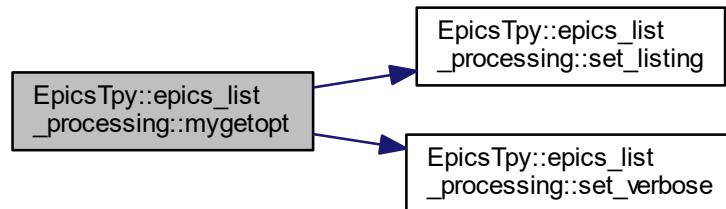
Number of arguments processed

Definition at line 587 of file TpyToEpics.cpp.

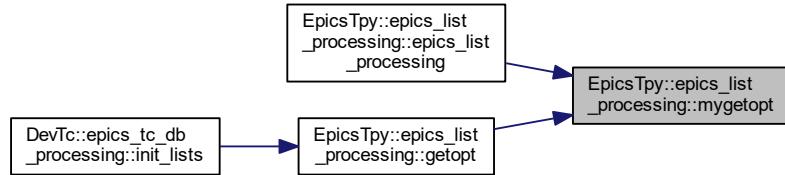
References EpicsTpy::listing_autoburt, EpicsTpy::listing_daqini, EpicsTpy::listing_standard, set_listing(), and set_verbose().

Referenced by epics_list_processing(), and getopt().

Here is the call graph for this function:



Here is the caller graph for this function:

**8.25.3.3 operator()()**

```
bool EpicsTpy::epics_list_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

Parameters

<code>arg</code>	Process argument describign the variable and type
------------------	---

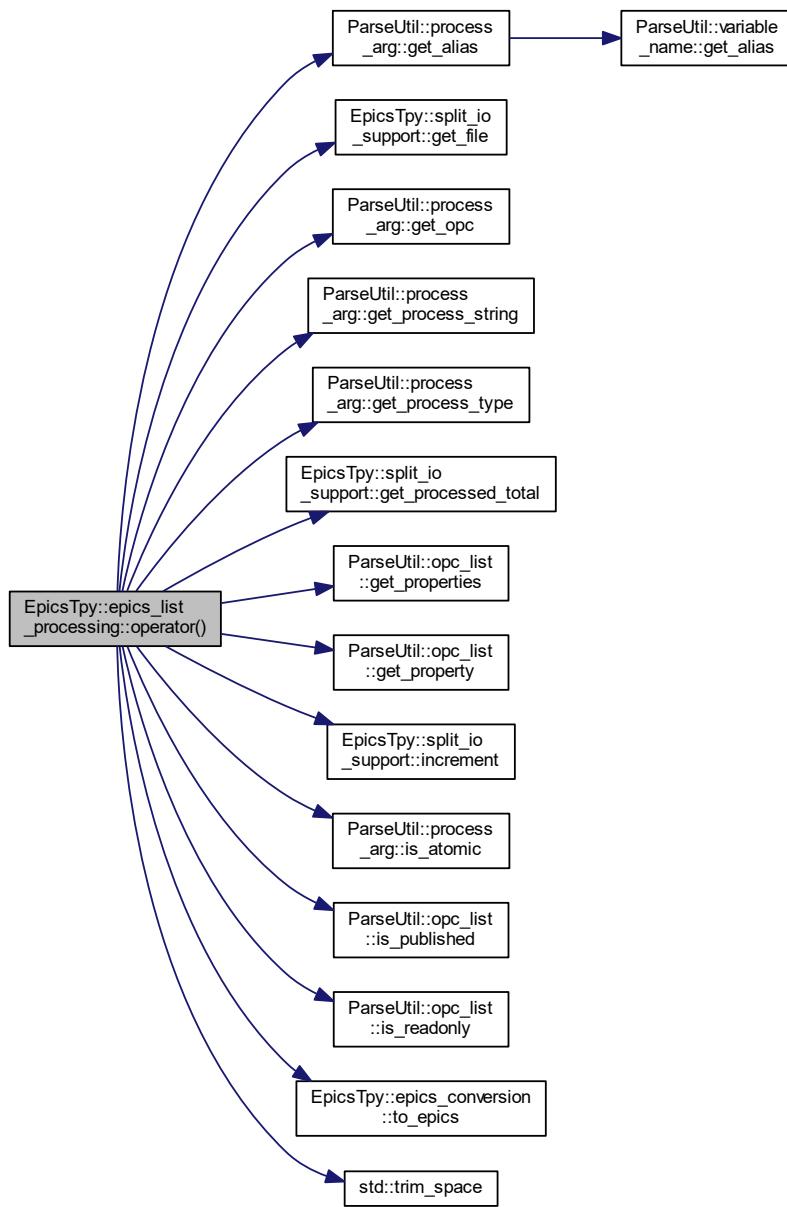
Returns

True if successfully processed

Definition at line 634 of file TpyToEpics.cpp.

References `ParseUtil::process_arg::get_alias()`, `EpicsTpy::split_io_support::get_file()`, `ParseUtil::process_arg::get_opc()`, `ParseUtil::process_arg::get_process_string()`, `ParseUtil::process_arg::get_process_type()`, `EpicsTpy::split_io_support::get_processed_total()`, `ParseUtil::opc_list::get_properties()`, `ParseUtil::opc_list::get_property()`, `EpicsTpy::split_io_support::increment()`, `ParseUtil::process_arg::is_atomic()`, `ParseUtil::opc_list::is_published()`, `ParseUtil::opc_list::is_READONLY()`, `EpicsTpy::LIGODAQ_DATATYPE_DEFAULT`, `EpicsTpy::LIGODAQ_DATATYPE_FLOAT`, `EpicsTpy::LIGODAQ_DATATYPE_INT32`, `EpicsTpy::LIGODAQ_DATATYPE_NAME`, `EpicsTpy::LIGODAQ_INI_HEADER`, `EpicsTpy::LIGODAQ_UNIT_DEFAULT`, `EpicsTpy::LIGODAQ_UNIT_NAME`, `EpicsTpy::LIGODAQ_UNIT_NONE`, `listing`, `EpicsTpy::listing_autoburt`, `EpicsTpy::listing_daqini`, `EpicsTpy::listing_standard`, `ParseUtil::OPC_PROP_CLOSE`, `ParseUtil::OPC_PROP_FFST`, `ParseUtil::OPC_PROP_OPEN`, `ParseUtil::OPC_PROP_UNIT`, `ParseUtil::OPC_PROP_ZRST`, `ParseUtil::pt_bool`, `ParseUtil::pt_enum`, `ParseUtil::pt_int`, `EpicsTpy::epics_conversion::to_epics()`, `std::trim_space()`, and `verbose`.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

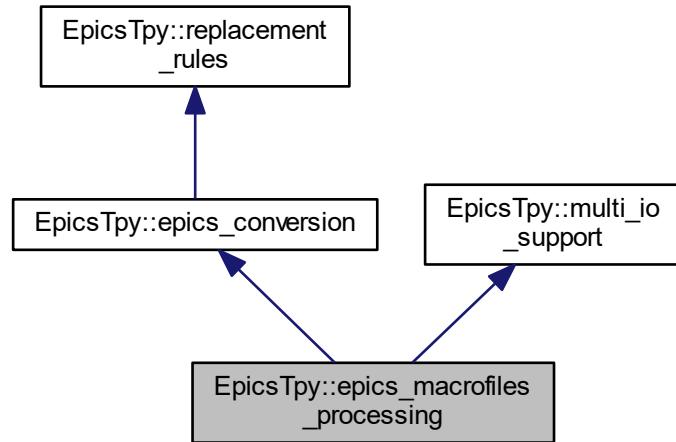
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.26 EpicsTpy::epics_macrofiles_processing Class Reference

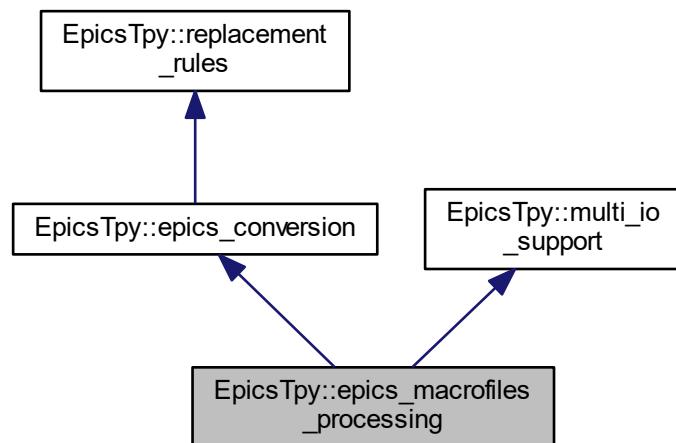
Macro file processing.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::epics_macrofiles_processing:



Collaboration diagram for EpicsTpy::epics_macrofiles_processing:



Public Member Functions

- [epics_macrofiles_processing \(\)](#)
Default constructor.
- [epics_macrofiles_processing \(macrofile_type mt\)](#)

- `epics_macrofiles_processing` (const `std::stringcase` &pname, const `std::stringcase` &dbname, bool tcat3, int argc, const char *const argv[], bool argp[] = 0)
- `~epics_macrofiles_processing` ()
Destructor.
- void `flush` ()
flush all pending processing
- int `getopt` (int argc, const char *const argv[], bool argp[] = 0)
- int `my getopt` (int argc, const char *const argv[], bool argp[] = 0)
- bool `operator()` (const `ParseUtil::process_arg` &arg)
- `macrofile_type get_macrofile_type` () const
Get listing type.
- void `set_macrofile_type` (`macrofile_type` m)
Set listing.
- void `set_plcname` (const `std::stringcase` &name)
Set PLC name.
- const `std::stringcase` & `get_plcname` () const
Get PLC name.
- bool `is_twincat3` () const
Is this twincat 3?
- void `set_twincat3` (bool tcat3 = true)
Set twincat 3 version?
- `std::stringcase to_filename` (const `std::stringcase` &epicsname)
Translate epics name to filename.
- int `get_processed_total` () const
Get number of processed channels.

Static Public Attributes

- static const `std::stringcase` `errorstruct` = "ErrorStruct"
Type name identifying an error struct ("ErrorStruct")
- static const `std::stringcase` `errorlistext2` = "_Errors.exp"
TwinCAT 2.11: File extension identifying a list of error msgs ("_Errors.exp")
- static const `std::regex` `errormatchregex2`
TwinCAT 2.11: Regular expression to match the entire error record definition.
- static const `std::stringcase` `errorlistext31` = "_Errors.TcGVL"
TwinCAT 3.1: File extension identifying a list of error msgs ("_Errors.TcGVL")
- static const `std::regex` `errormatchregex31`
TwinCAT 3.1: Regular expression to match the entire error record definition.
- static const `std::regex` `errorsearchregex`
TwinCAT 2.11/3.1: Regular expression to search for the actual error messages.

Protected Member Functions

- bool `process_record` (const `macro_record` &mrec, int level = 0)
Process top of stack.

Protected Attributes

- `macrofile_type macros`
Listing type.
- `std::stringcase plcname`
PLC name.
- `bool isTwinCAT3`
TwinCAT version.
- `macro_stack procstack`
Processing stack.
- `int rec_num`
Current number of processed channels (records)
- `filename_set missing`
set of missing input files

8.26.1 Detailed Description

Macro file processing.

Class for generating macro files to be used by medm

Definition at line 612 of file TpyToEpics.h.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 epics_macrofiles_processing() [1/2]

```
EpicsTpy::epics_macrofiles_processing::epics_macrofiles_processing (
    macrofile_type mt )  [inline], [explicit]
```

Constructor

Parameters

<code>mt</code>	Type of macro
-----------------	---------------

Definition at line 632 of file TpyToEpics.h.

8.26.2.2 epics_macrofiles_processing() [2/2]

```
EpicsTpy::epics_macrofiles_processing::epics_macrofiles_processing (
    const std::stringcase & pname,
    const std::stringcase & dname,
```

```

    bool tcat3,
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } )

```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored Processed options with [epics_conversion::getopt](#), [multi_io_support::getopt](#) and [my getopt\(\)](#).

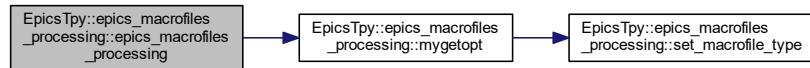
Parameters

<i>pname</i>	PLC name
<i>dname</i>	Directory name
<i>tcat3</i>	True if we are processing TwinCAT 3.1 files
<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 739 of file TpyToEpics.cpp.

References EpicsTpy::all, and my getopt().

Here is the call graph for this function:



8.26.3 Member Function Documentation

8.26.3.1 getopt()

```

int EpicsTpy::epics_macrofiles_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = { 0 } )

```

Parse a command line Processed options with [epics_conversion::getopt](#) and [my getopt\(\)](#).

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

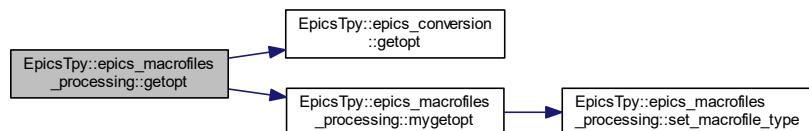
Returns

Number of arguments processed

Definition at line 764 of file TpyToEpics.cpp.

References EpicsTpy::epics_conversion::getopt(), and my getopt().

Here is the call graph for this function:

**8.26.3.2 my getopt()**

```
int EpicsTpy::epics_macrofiles_processing::my getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/mf: Generate a macro file for each structure describing all fields /me: Generate a macro file for each structure describing the error messages /ma: Generate a macro file for each structure describing fields and errors (default)

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

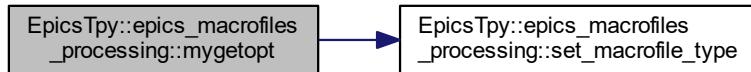
Number of arguments processed

Definition at line 774 of file TpyToEpics.cpp.

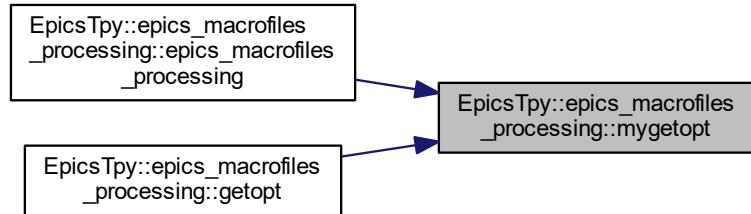
References EpicsTpy::all, EpicsTpy::errors, EpicsTpy::fields, and set_macrofile_type().

Referenced by epics_macrofiles_processing(), and getopt().

Here is the call graph for this function:



Here is the caller graph for this function:



8.26.3.3 operator()()

```
bool EpicsTpy::epics_macrofiles_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

Parameters

<i>arg</i>	Process argument describign the variable and type
------------	---

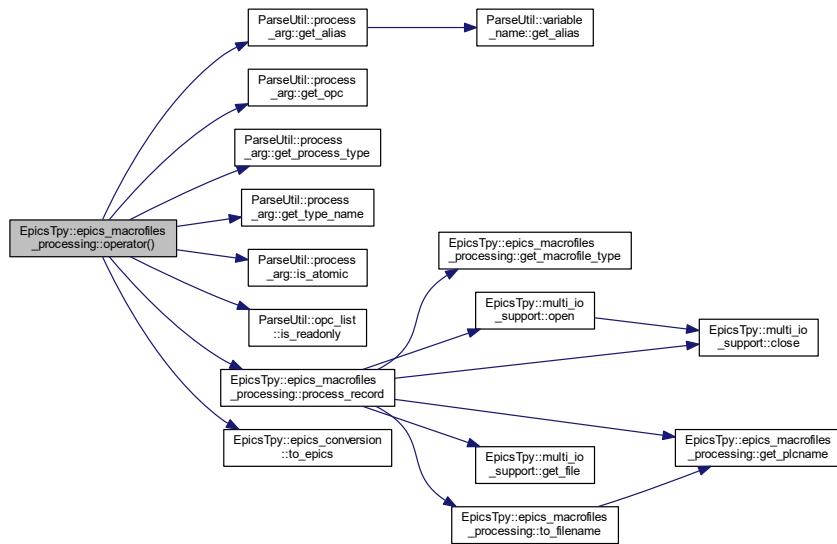
Returns

True if successfully processed

Definition at line 812 of file TpyToEpics.cpp.

References EpicsTpy::macro_record::back, errorstruct, ParseUtil::process_arg::get_alias(), ParseUtil::process_arg::get_opc(), ParseUtil::process_arg::get_process_type(), ParseUtil::process_arg::get_type_name(), ParseUtil::process_arg::is_atomic(), ParseUtil::opc_list::is_READONLY(), EpicsTpy::macro_record::iserror, EpicsTpy::macro_info::name, process_record(), procstack, ParseUtil::pt_INVALID, EpicsTpy::macro_info::ptype, EpicsTpy::macro_info::readonly, rec_num, EpicsTpy::macro_record::record, EpicsTpy::epics_conversion::to_epics(), and EpicsTpy::macro_info::type_n.

Here is the call graph for this function:



8.26.3.4 to_filename()

```
std::stringcase EpicsTpy::epics_macros_processing::to_filename (
    const std::stringcase & epicsname )
```

Translate epics name to filename.

Translate epics name to filename

Definition at line 875 of file TpyToEpics.cpp.

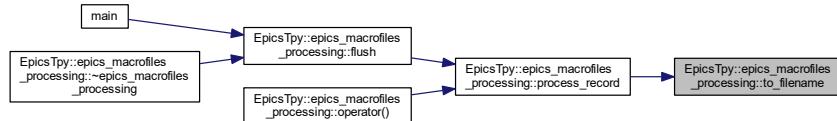
References `get_plcname()`, and `isTwinCAT3`.

Referenced by `process_record()`.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.27 DevTc::epics_record_traits< RecType > Struct Template Reference

Epics record traits.

```
#include <devTc.h>
```

Classes

- struct `traits_type`

Epics record type.

Public Types

- `typedef epicsFloat64 value_type`

Value type of (raw) value field.

Static Public Member Functions

- `static const char *const name ()`

Name of the record.

- `static value_type * val (traits_type *prec)`

Returns the (raw) value of a record.

- `static bool read (traits_type *epicsrec, plc::BaseRecord *baserec)`

Performs the read access on prec.

- `static bool write (plc::BaseRecord *baserec, traits_type *epicsrec)`

Performs the write access on prec.

Static Public Attributes

- static const aitEnum `value_ait_type` = aitEnumFloat64
- static const aitInt32 `value_count` = 0
- static const int `value_conversion` = 0
return value for read_io functions 0=default, 2=don't convert
- static const bool `input_record` = true
Indicates if this is an input record.
- static const bool `raw_record` = false
Indicates if this is a raw record.

8.27.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::epics_record_traits< RecType >
```

Epics record traits.

This traits class for Epics records.

Definition at line 259 of file devTc.h.

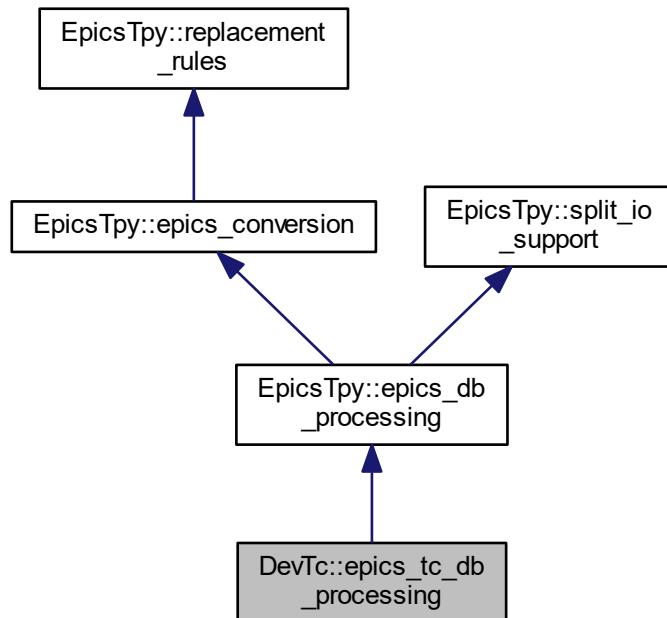
The documentation for this struct was generated from the following file:

- [devTc.h](#)

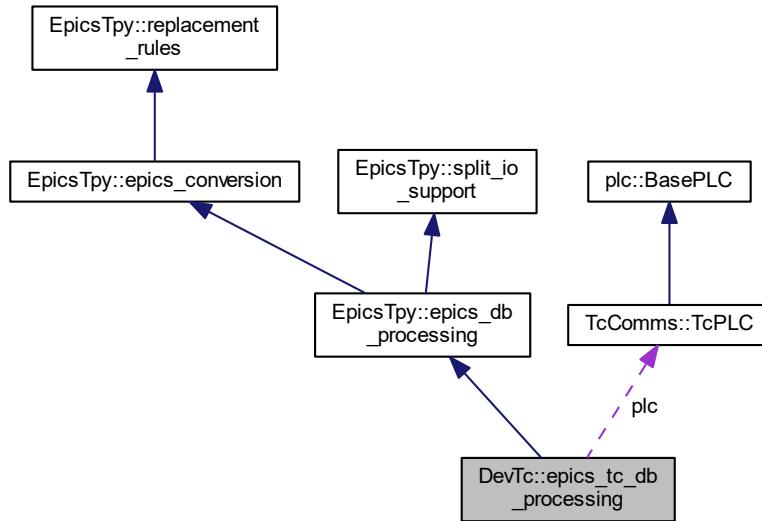
8.28 DevTc::epics_tc_db_processing Class Reference

EPICS/TCat db processing.

Inheritance diagram for DevTc::epics_tc_db_processing:



Collaboration diagram for DevTc::epics_tc_db_processing:



Public Member Functions

- `epics_tc_db_processing (TcComms::TcPLC &p, EpicsTpy::replacement_table &rules, tc_listing_def *l=nullptr, tc_macro_def *m=nullptr)`
Default constructor.
- `bool operator() (const ParseUtil::process_arg &arg)`
- `void flush ()`
Flush output files.
- `int get_invalid_records () const`
Get number of EPICS records without tc records.

Protected Member Functions

- `epics_tc_db_processing (const epics_tc_db_processing &)`
Disable copy constructor.
- `epics_tc_db_processing & operator= (const epics_tc_db_processing &)`
Disable assignment operator.
- `void init_lists ()`
Init lists.
- `void done_lists ()`
Cleanup lists.
- `bool process_lists (const ParseUtil::process_arg &arg)`
- `bool process_list (filename_rule_list_tuple &listdef, const ParseUtil::process_arg &arg)`
- `void init_macros ()`
Init macros.
- `void done_macros ()`
Cleanup macros.
- `bool process_macros (const ParseUtil::process_arg &arg)`
- `bool process_macro (dirname_arg_macro_tuple ¯odef, const ParseUtil::process_arg &arg)`

Protected Attributes

- `TcComms::TcPLC * plc`
Pointer to PLC class.
- `tc_listing_def * lists`
Pointer to a set of listings.
- `tc_macro_def * macros`
Pointer to macros.
- `int invnum`
Number of EPICS records without tc records.

Additional Inherited Members

8.28.1 Detailed Description

EPICS/TCat db processing.

Class for generating an EPICS database and tc record

Definition at line 85 of file drvTc.cpp.

8.28.2 Member Function Documentation

8.28.2.1 operator()()

```
bool DevTc::epics_tc_db_processing::operator() (
    const ParseUtil::process_arg & arg )
```

Process a variable

Parameters

<code>arg</code>	Process argument describign the variable and type
------------------	---

Returns

True if successful

Determine data type of this record

Make new record object

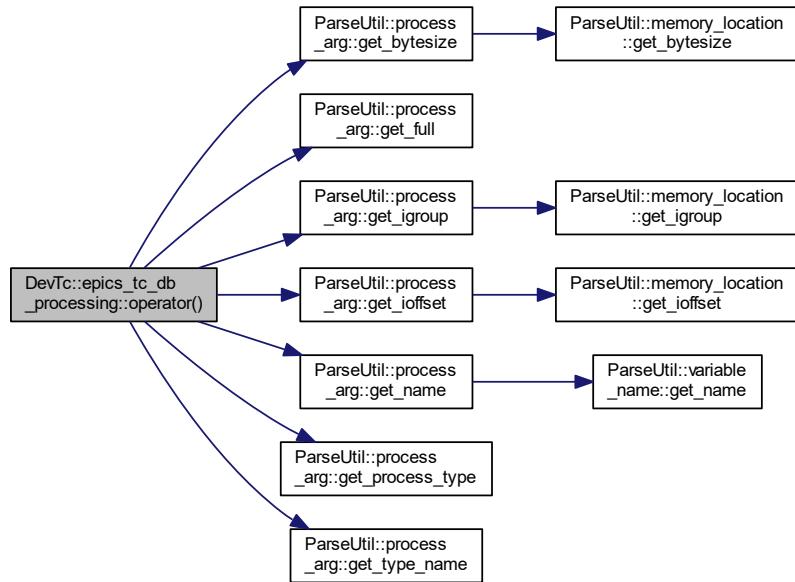
Make TCat interface

Tell record about TCat interface

Definition at line 300 of file drvTc.cpp.

References `plc::dtBool`, `plc::dtDouble`, `plc::dtFloat`, `plc::dtInt16`, `plc::dtInt32`, `plc::dtInt8`, `plc::dtInvalid`, `plc::dtString`, `plc::dtUInt16`, `plc::dtUInt32`, `plc::dtUInt8`, `ParseUtil::process_arg::get_bytesize()`, `ParseUtil::process_arg::get_full()`, `ParseUtil::process_arg::get_igroup()`, `ParseUtil::process_arg::get_ioffset()`, `ParseUtil::process_arg::get_name()`, `ParseUtil::process_arg::get_process_type()`, `ParseUtil::process_arg::get_type_name()`, `ParseUtil::pt_binary`, and `ParseUtil::pt_enum`.

Here is the call graph for this function:



8.28.2.2 process_list()

```
bool DevTc::epics_tc_db_processing::process_list (
    filename_rule_list_tuple & listdef,
    const ParseUtil::process_arg & arg ) [protected]
```

Process a listing

Parameters

<code>listdef</code>	filename/rule pair defining a listing
<code>arg</code>	Process argument describign the variable and type

Returns

True if successful

Definition at line 228 of file drvTc.cpp.

References ParseUtil::process_arg::get_process_type(), and ParseUtil::pt_string.

Here is the call graph for this function:



8.28.2.3 process_lists()

```
bool DevTc::epics_tc_db_processing::process_lists (
    const ParseUtil::process_arg & arg ) [protected]
```

Process all listings

Parameters

<i>arg</i>	Process argument describign the variable and type
------------	---

Returns

True if successful

Definition at line 215 of file drvTc.cpp.

8.28.2.4 process_macro()

```
bool DevTc::epics_tc_db_processing::process_macro (
    dirname_arg_macro_tuple & macrodef,
    const ParseUtil::process_arg & arg ) [protected]
```

Process a macro

Parameters

<i>macrodef</i>	filename/rule pair defining a macro
<i>arg</i>	Process argument describign the variable and type

Returns

True if successful

Definition at line 289 of file drvTc.cpp.

8.28.2.5 process_macros()

```
bool DevTc::epics_tc_db_processing::process_macros (
    const ParseUtil::process_arg & arg ) [protected]
```

Process all macros

Parameters

<code>arg</code>	Process argument describign the variable and type
------------------	---

Returns

True if successful

Definition at line 276 of file drvTc.cpp.

The documentation for this class was generated from the following file:

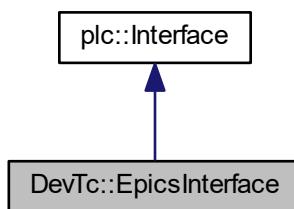
- [drvTc.cpp](#)

8.29 DevTc::EpicsInterface Class Reference

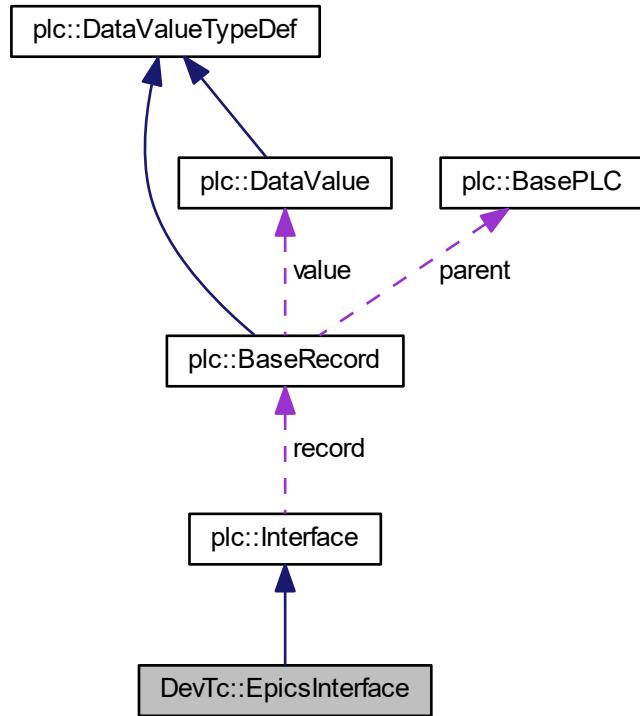
Epics interface class.

```
#include <devTc.h>
```

Inheritance diagram for DevTc::EpicsInterface:



Collaboration diagram for DevTc::EpicsInterface:



Public Member Functions

- `EpicsInterface (plc::BaseRecord &dval)`
Constructor.
- `~EpicsInterface ()`
Deconstructor.
- `void set_isPassive (bool passive)`
Set isPassive.
- `bool get_isCallback () const`
Get isCallback.
- `void set_isCallback (bool isCb)`
Set isCallback.
- `void set_pEpicsRecord (dbCommon *pEpRecord)`
Set pEpicsRecord.
- `void *get_pEpicsVal () const`
Get pEpicsVal.
- `void set_pEpicsVal (void *pVal)`
Set pEpicsVal.
- `unsigned long get_size () const`
Get size.
- `void set_size (unsigned long nBytes)`

- Set size.
- bool `get_callbackRequestPending () const`
Get callbackRequestPending.
- const CALLBACK & `callback () const`
Get pointer to callback structure.
- CALLBACK & `callback ()`
Get pointer to callback structure.
- const IOSCANPVT & `ioscan () const`
Get reference to io scan list pointer.
- IOSCANPVT & `ioscan ()`
Get reference to io scan list pointer.
- IOSCANPVT `get_ioscan () const`
Get pointer to io scan list.
- void `set_ioscan (const IOSCANPVT ioscan)`
Set pointer to io scan list.
- virtual bool `push () override`
Makes a call to the EPICS dbProcess function.
- virtual bool `pull () override`
Does nothing.

Protected Attributes

- bool `isPassive`
- bool `isCallback`
- dbCommon * `pEpicsRecord`
Pointer to the EPICS record.
- void * `pEpicsVal`
Pointer to the RVAL field of the EPICS record.
- unsigned long `size`
Size (bytes) of the data.
- IOSCANPVT `ioscanpvt`
Pointer to IO scan list.
- CALLBACK `callbackval`
Callback structure.

8.29.1 Detailed Description

Epics interface class.

This is a class for an EPICS Interface

Definition at line 110 of file devTc.h.

The documentation for this class was generated from the following files:

- `devTc.h`
- `devTc.cpp`

8.30 std::std::hash< std::stringcase > Struct Template Reference

hash for case insensitive string.

```
#include <stringcase_hash.h>
```

Public Member Functions

- std::size_t **operator()** (const stringcase &str) const

8.30.1 Detailed Description

```
template<>
struct std::std::hash< std::stringcase >
```

hash for case insensitive string.

This is a function specialization for case sensitive strings. Perform a 32/64 bit Fowler/Noll/Vo hash on a string.

Definition at line 15 of file stringcase_hash.h.

8.30.2 Member Function Documentation

8.30.2.1 operator()()

```
std::size_t std::std::hash< std::stringcase >::operator() (
    const stringcase & str ) const
```

Perform a 32 or 64 bit Fowler/Noll/Vo hash on a string Adapted from fnv_32_str and fnv_64_str <http://www.isthe.com/chongo/tech/comp/fnv/index.html#FNV-source>

Parameters

<i>str</i>	string to hash
------------	----------------

Returns

32/64 bit hash

The documentation for this struct was generated from the following file:

- [stringcase_hash.h](#)

8.31 std::std::hash< std::wstringcase > Struct Template Reference

hash for case insensitive unicode string.

```
#include <stringcase_hash.h>
```

Public Member Functions

- std::size_t [operator\(\)](#) (const `wstringcase` &`str`) const

8.31.1 Detailed Description

```
template<>
struct std::std::hash< std::wstringcase >
```

hash for case insensitive unicode string.

This is a function specialization for case sensitive unicode strings. Perform a 32/64 bit Fowler/Noll/Vo hash on a unicode string.

Definition at line 31 of file `stringcase_hash.h`.

8.31.2 Member Function Documentation

8.31.2.1 operator()()

```
std::size_t std::std::hash< std::wstringcase >::operator() (
    const wstringcase & str ) const
```

Perform a 32 or 64 bit Fowler/Noll/Vo hash on a string Adapted from `fnv_32_str` <http://www.isthe.com/chongo/tech/comp/fnv/index.html#FNV-source>

Parameters

<code>str</code>	string to hash
------------------	----------------

Returns

32/64 bit hash

The documentation for this struct was generated from the following file:

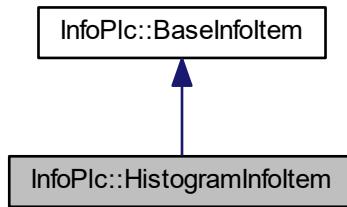
- [stringcase_hash.h](#)

8.32 InfoPlc::HistogramInfoItem Class Reference

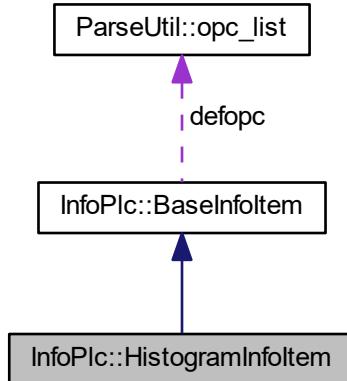
Histogram statistic.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::HistogramInfoItem:



Collaboration diagram for InfoPlc::HistogramInfoItem:



Public Member Functions

- [HistogramInfoItem \(\)](#)
Default constructor.
- [~HistogramInfoItem \(\)](#)
Destructor.
- virtual void [update \(double val\)](#) override
Update.
- virtual void [reset \(\)](#) override
Reset.

Protected Attributes

- double `ulim`
Upper limit.
- double `llim`
Lower limit.
- double `nBuckets`
Number of buckets.
- int `cnt`
Number of data points.
- std::list< int > `buckets`
Histogram.

8.32.1 Detailed Description

Histogram statistic.

This is a class for a histogram

Definition at line 142 of file `infoPlc.h`.

The documentation for this class was generated from the following file:

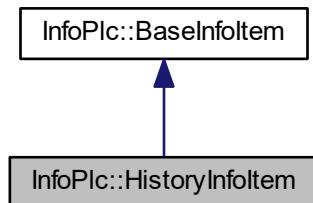
- [infoPlc.h](#)

8.33 InfoPlc::HistoryInfoltem Class Reference

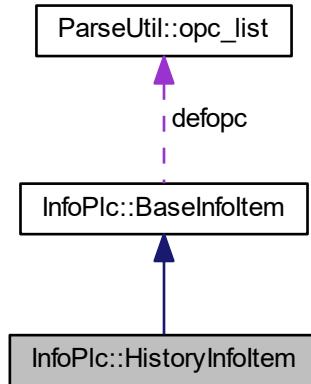
History tracker.

```
#include <infoPlc.h>
```

Inheritance diagram for `InfoPlc::HistoryInfoltem`:



Collaboration diagram for InfoPlc::HistoryInfoltem:



Public Member Functions

- `HistoryInfoltem` (int length=10)
Default constructor.
- `HistoryInfoltem` (const `ParseUtil::opc_list` &opc, int length=10)
Constructor from opc list.
- `~HistoryInfoltem` ()
Destructor.
- virtual bool `setup` (const `std::stringcase` &tagname, const `std::stringcase` &plc_alias, `plc::Interface` *puser) override
- virtual void `update` (double val) override
Update.
- virtual void `reset` () override
Reset.

Protected Attributes

- int `history_length`
Length of history to keep.
- `std::list< double > history`
History of values.

8.33.1 Detailed Description

History tracker.

This is a class for keeping the history of a something

Definition at line 169 of file `infoPlc.h`.

8.33.2 Member Function Documentation

8.33.2.1 setup()

```
bool InfoPlc::HistoryInfoItem::setup (
    const std::string& tagname,
    const std::string& plc_alias,
    plc::Interface * puser ) [override], [virtual]
```

Setup

Parameters

<i>tagname</i>	Name of channels
<i>plc_alias</i>	PLC alias name
<i>puser</i>	Pointer ot PLC

HistoryInfoItem

Implements [InfoPlc::BaseInfoItem](#).

Definition at line 121 of file [infoPlc.cpp](#).

The documentation for this class was generated from the following files:

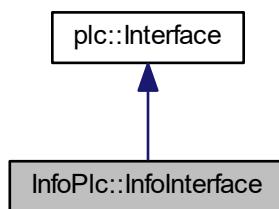
- [infoPlc.h](#)
- [infoPlc.cpp](#)

8.34 InfoPlc::InfoInterface Class Reference

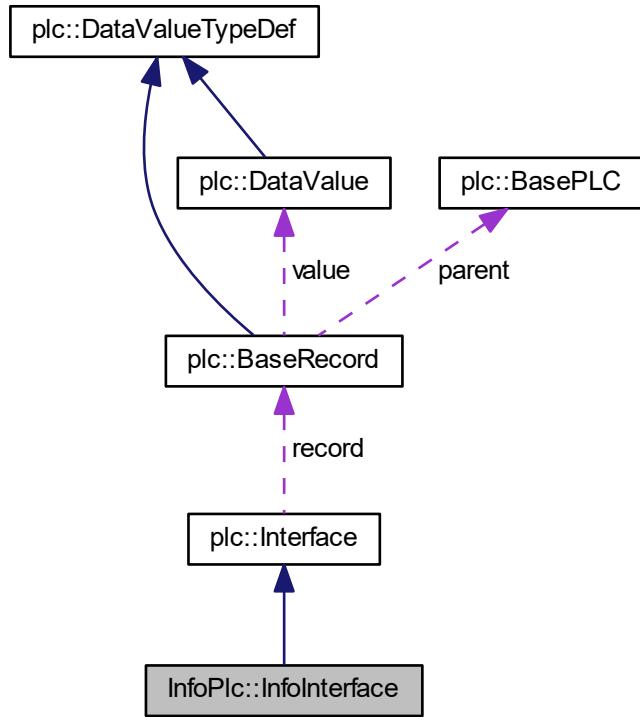
Info interface.

```
#include <infoPlc.h>
```

Inheritance diagram for [InfoPlc::InfoInterface](#):



Collaboration diagram for InfoPlc::InfoInterface:



Public Member Functions

- `InfoInterface (plc::BaseRecord &dval, std::stringcase statname)`
Constructor.
- `~InfoInterface ()`
Destructor.
- `virtual bool push () override`
push data
- `virtual bool pull () override`
pull data

Protected Attributes

- `std::stringcase name`
Name of statistics.

8.34.1 Detailed Description

Info interface.

This is a class for a Info interface

Definition at line 224 of file infoPlc.h.

The documentation for this class was generated from the following file:

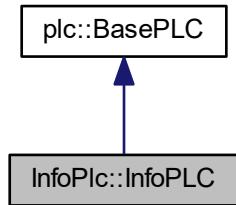
- [infoPlc.h](#)

8.35 InfoPlc::InfoPLC Class Reference

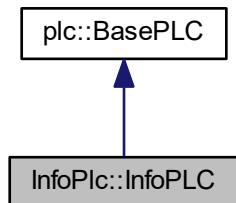
Info plc.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::InfoPLC:



Collaboration diagram for InfoPlc::InfoPLC:



Public Member Functions

- template<class Function >
int **process_info** (Function &process, const std::stringcase &prefix=std::stringcase()) const
Process the type tree of a symbol.
- const std::stringcase & **get_prefix** () const
get the tag prefix for the info PLC
- void **set_prefix** (const std::stringcase &pre)
get the tag prefix for the info PLC

Protected Member Functions

- template<class Function >
int **process_info** (const BaseInfoItem &info, Function &process, const ParseUtil::opc_list &defopc, ParseUtil::memory_location &memloc, const std::stringcase &prefix=std::stringcase()) const
Process the type tree of a symbol.

Protected Attributes

- BaseInfoList info_list**
List of statistics values.
- std::stringcase prefix**
Tag prefix.

Additional Inherited Members

8.35.1 Detailed Description

Info plc.

This is a class for a dummy PLC that tracks connection status, errors, etc.

Definition at line 247 of file infoPlc.h.

8.35.2 Member Function Documentation

8.35.2.1 process_info() [1/2]

```
template<class Function >
int InfoPlc::InfoPLC::process_info (
    Function & process,
    const std::stringcase & prefix = std::stringcase() ) const
```

Process the type tree of a symbol.

Iterates over the info list and processes all specified tags.

Parameters

<i>process</i>	Function class
<i>prefix</i>	Prefix which is added to all variable names

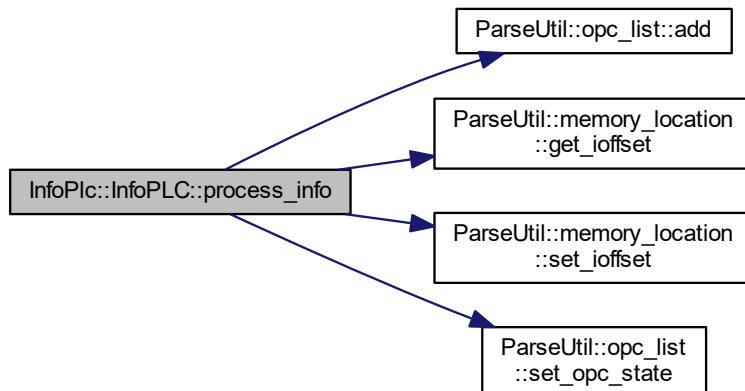
Returns

Number of processes variables

Definition at line 9 of file infoPlcTemplate.h.

References ParseUtil::opc_list::add(), ParseUtil::memory_location::get_ioffset(), info_list, ParseUtil::OPC_PROP←_PLCNAME, prefix, ParseUtil::publish, ParseUtil::memory_location::set_ioffset(), and ParseUtil::opc_list::set_opc←_state().

Here is the call graph for this function:

**8.35.2.2 process_info() [2/2]**

```

template<class Function >
int InfoPlc::InfoPLC::process_info (
    const BaseInfoItem & info,
    Function & process,
    const ParseUtil::opc_list & defopc,
    ParseUtil::memory_location & memloc,
    const std::stringcase & prefix = std::stringcase() ) const [protected]
  
```

Process the type tree of a symbol.

Iterates over the info list and processes all specified tags.

Parameters

<i>info</i>	Info item
<i>process</i>	Function class
<i>defopc</i>	OPC default list
<i>memloc</i>	memory location used to get the EPICS name
<i>prefix</i>	Prefix which is added to all variable names

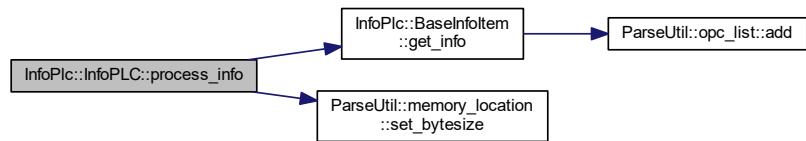
Returns

Number of processes variables

Definition at line 27 of file infoPlcTemplate.h.

References InfoPlc::BaseInfoItem::get_info(), plc::BasePLC::name, prefix, and ParseUtil::memory_location::set_bytesize().

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- [infoPlc.h](#)
- [infoPlcTemplate.h](#)

8.36 DevInfo::InfoRegisterTolocShell Class Reference

Register info commands.

```
#include <drvInfo.h>
```

8.36.1 Detailed Description

Register info commands.

Register Info commands to IOC shell This class registers the callback functions for the Info IOC commands

Definition at line 18 of file drvInfo.h.

The documentation for this class was generated from the following files:

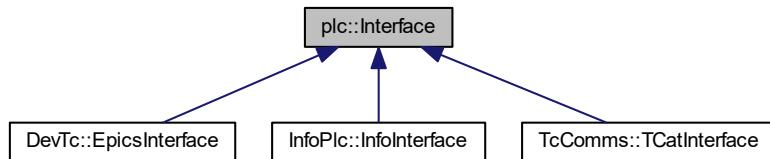
- [drvInfo.h](#)
- [drvInfo.cpp](#)

8.37 plc::Interface Class Reference

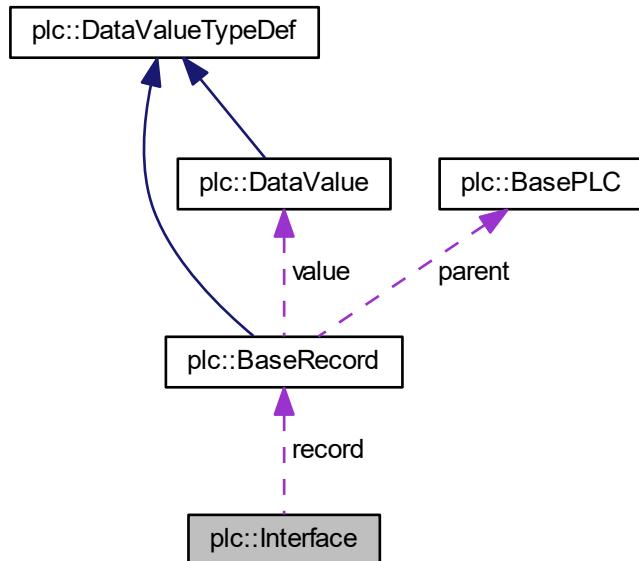
Abstract interface.

```
#include <plcBase.h>
```

Inheritance diagram for plc::Interface:



Collaboration diagram for plc::Interface:



Public Member Functions

- `Interface (BaseRecord &dval)`
- `virtual ~Interface ()`
Destructor.
- `BaseRecord * get_record () const`

Return a pointer to the tag/channel record.

- void [set_record \(BaseRecord &rec\)](#)

Set the tag/channel record reference.

- virtual bool [push \(\)=0](#)

Pure virtual method indicating that the value needs to be pushed.

- virtual bool [pull \(\)=0](#)

Pure virtual method indicating that the value needs to be pulled.

Protected Attributes

- [BaseRecord * record](#)

Pointer to tag/channel record associated with this interface.

8.37.1 Detailed Description

Abstract interface.

This is a base class for an abstract interface to access the PLC (slave) or the user side (master).

Definition at line 33 of file plcBase.h.

8.37.2 Constructor & Destructor Documentation

8.37.2.1 Interface()

```
plc::Interface::Interface (
    BaseRecord & dval ) [inline], [explicit]
```

Constructor

Parameters

dval	Reference to a tag/channel record
----------------------	-----------------------------------

Definition at line 38 of file plcBase.h.

The documentation for this class was generated from the following file:

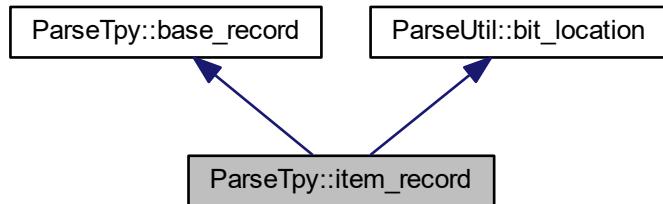
- [plcBase.h](#)

8.38 ParseTpy::item_record Class Reference

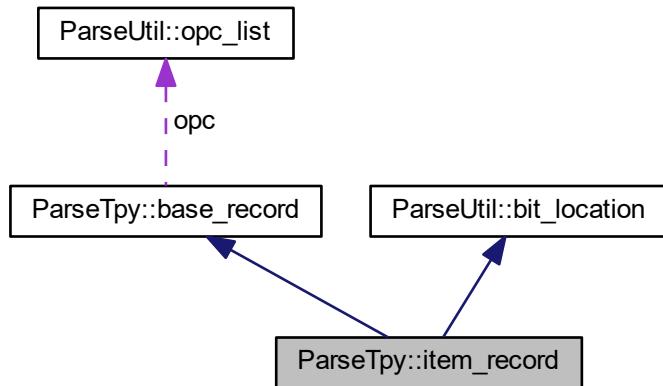
item record

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::item_record:



Collaboration diagram for ParseTpy::item_record:



Public Member Functions

- [item_record \(\)](#)

Default constructor.

Additional Inherited Members

8.38.1 Detailed Description

item record

This class stores typed items.

Definition at line 234 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.39 EpicsTpy::macro_info Struct Reference

Macro information.

```
#include <TpyToEpics.h>
```

Public Member Functions

- [macro_info \(\)](#)
Default constructor.

Public Attributes

- [ParseUtil::process_type_enum ptype](#)
Process Type.
- [std::stringcase name](#)
name of type
- [std::stringcase type_n](#)
type definition
- [bool readonly](#)
readonly

8.39.1 Detailed Description

Macro information.

This structure describes a field

Definition at line 564 of file TpyToEpics.h.

The documentation for this struct was generated from the following file:

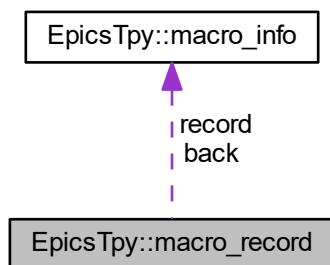
- [TpyToEpics.h](#)

8.40 EpicsTpy::macro_record Struct Reference

Macro record.

```
#include <TpyToEpics.h>
```

Collaboration diagram for EpicsTpy::macro_record:



Public Attributes

- [macro_info record](#)
name of structure
- [bool iserror](#)
is an ErrorStruct
- [bool haserror](#)
contains an ErrorStruct
- [int erroridx](#)
index if fields list to ErrorStruct
- [macro_list fields](#)
List of fields.
- [macro_info back](#)
name of upper level structure

8.40.1 Detailed Description

Macro record.

This structure describes a record/struct

Definition at line 585 of file TpyToEpics.h.

The documentation for this struct was generated from the following file:

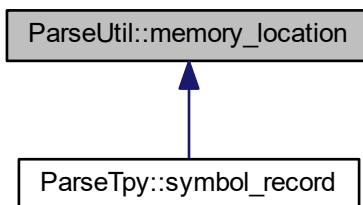
- [TpyToEpics.h](#)

8.41 ParseUtil::memory_location Class Reference

Memory location.

```
#include <ParseUtil.h>
```

Inheritance diagram for ParseUtil::memory_location:



Public Member Functions

- `memory_location ()`
Default constructor.
- `memory_location (int ig, int io, int bs)`
- `memory_location (const std::stringcase &s)`
- `bool isValid () const`
Validity.
- `int get_igroup () const`
Get IGroup.
- `void set_igroup (int ig)`
Set IGroup.
- `int get_ioffset () const`
Get IOffset.
- `void set_ioffset (int io)`
Set IOffset.
- `int get_bytesize () const`
Get BitSize.
- `void set_bytesize (int bs)`
Set BitSize.
- `bool set_section (const bit_location &loc)`
- `std::stringcase get () const`
- `bool set (const std::stringcase &s)`

Protected Attributes

- `int igrup`
Memory group.
- `int ioffset`
Memory offset.
- `int bytesize`
Memory size in bits.

8.41.1 Detailed Description

Memory location.

This structure holds a memory location

Definition at line 219 of file ParseUtil.h.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 `memory_location()` [1/2]

```
ParseUtil::memory_location::memory_location (
    int ig,
    int io,
    int bs )  [inline]
```

Constructor

Parameters

<i>ig</i>	Index group
<i>io</i>	Index offset
<i>bs</i>	Size in bytes

Definition at line 228 of file ParseUtil.h.

8.41.2.2 memory_location() [2/2]

```
ParseUtil::memory_location::memory_location (
    const std::stringcase & s ) [inline]
```

Constructor

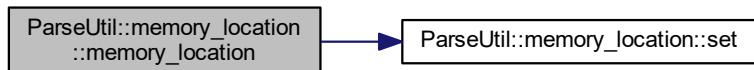
Parameters

<i>s</i>	Definition string
----------	-------------------

Definition at line 232 of file ParseUtil.h.

References set().

Here is the call graph for this function:



8.41.3 Member Function Documentation

8.41.3.1 get()

```
std::stringcase ParseUtil::memory_location::get ( ) const
```

Gets a string representation of a memory location

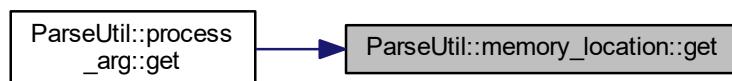
Returns

string with format "igroup/offset:size", empty on error

Definition at line 212 of file ParseUtil.cpp.

Referenced by ParseUtil::process_arg::get().

Here is the caller graph for this function:

**8.41.3.2 set()**

```
bool ParseUtil::memory_location::set (
    const std::string& s )
```

Set the memory location using a string of the form: "igroup/offset:size" where the size is in bytes

Parameters

<code>s</code>	String describing memory location
----------------	-----------------------------------

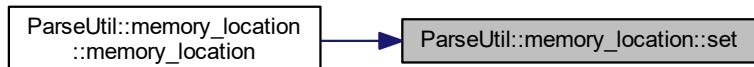
Returns

True if successful

Definition at line 225 of file ParseUtil.cpp.

Referenced by memory_location().

Here is the caller graph for this function:



8.41.3.3 `set_section()`

```
bool ParseUtil::memory_location::set_section (
    const bit_location & loc )
```

Set a sub section

Parameters

<i>loc</i>	Bit location within memory region
------------	-----------------------------------

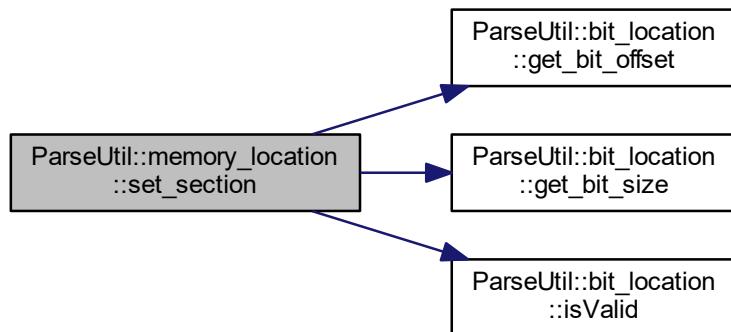
Returns

True if section within bounds, false otherwise

Definition at line 197 of file ParseUtil.cpp.

References `ParseUtil::bit_location::get_bit_offset()`, `ParseUtil::bit_location::get_bit_size()`, and `ParseUtil::bit_location::isValid()`.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

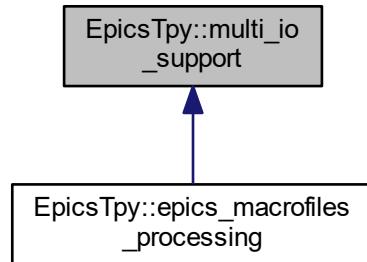
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.42 EpicsTpy::multi_io_support Class Reference

Multiple IO support.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::multi_io_support:



Public Member Functions

- `multi_io_support ()`
Default constructor.
- `multi_io_support (const std::stringcase &dbname)`
Constructor.
- `multi_io_support (const std::stringcase &dbname, int argc, const char *const argv[], bool argp[] = 0)`
- `~multi_io_support ()`
Destructor.
- `bool operator! () const`
Return error.
- `bool open (const std::stringcase &fname, const std::stringcase &io = "w", bool superrmsg = false)`
Open file for reading/writing.
- `void close ()`
Close file.
- `FILE * get_file () const`
Get file handle.
- `void set_outdirname (const std::stringcase &dbname)`
Set output directory name.
- `const std::stringcase & get_outdirname () const`
Get output directory name.
- `void set_indirname (const std::stringcase &dbname)`
Set input directory name.
- `const std::stringcase & get_indirname () const`
Get input directory name.
- `const std::stringcase & get_filename () const`
Get full filename.
- `io_filestat fileread () const`
Reading.
- `int get_filein_total () const`
Get number of read files.
- `int get_fileout_total () const`
Get number of written files.

Protected Member Functions

- void `set_filename` (const `std::stringcase` &fname)
Set output filename.
- int `getopt` (int argc, const char *const argv[], bool argp[] = 0)

Protected Attributes

- `std::stringcase outdirname`
Directory name.
- `std::stringcase indirname`
Directory name.
- `std::stringcase filename`
Current filename.
- `io_filestat filestat`
reading or writing?
- FILE * `filehandle`
Output file.
- int `file_num_in`
Current file number of processed read only channels (records)
- int `file_num_out`
Current file number of processed input/output channels (records)

8.42.1 Detailed Description

Multiple IO support.

Multi file IO support Supports a directory argument and opens files within

Definition at line 355 of file TpyToEpics.h.

8.42.2 Constructor & Destructor Documentation

8.42.2.1 multi_io_support()

```
EpicsTpy::multi_io_support::multi_io_support (
    const std::stringcase & dname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

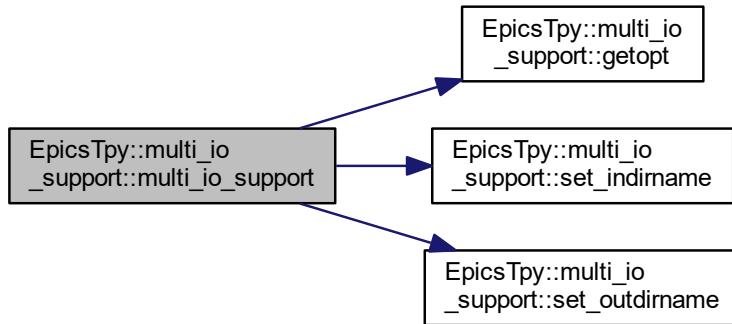
Parameters

<code>dname</code>	Name of output directory
<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in main()
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 374 of file TpyToEpics.h.

References getopt(), set_indirname(), and set_outdirname().

Here is the call graph for this function:



8.42.3 Member Function Documentation

8.42.3.1 getopt()

```
int EpicsTpy::multi_io_support::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline], [protected]
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

Number of arguments processed

Definition at line 433 of file TpyToEpics.h.

Referenced by multi_io_support().

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.43 ParseUtil::opc_list Class Reference

OPC list.

```
#include <ParseUtil.h>
```

Public Member Functions

- [`opc_list \(\)`](#)
Default constructor.
- [`opc_enum get_opc_state \(\) const`](#)
Get opc state.
- [`void set_opc_state \(opc_enum state\)`](#)
Set opc state.
- [`const property_map & get_properties \(\) const`](#)
Get opc property list.
- [`property_map & get_properties \(\)`](#)
Get opc property list.
- [`const property_map::const_iterator get_property \(int key\) const`](#)
Get the specified property.
- [`void add \(const property_el &el\)`](#)
Add an OPC property.
- [`void add \(const opc_list &o\)`](#)
Add an OPC list.
- [`bool is_published \(\) const`](#)
Is this item published?
- [`bool is_READONLY \(\) const`](#)
Is readonly?
- [`bool get_property \(int prop, std::stringcase &val\) const`](#)
Get string property.
- [`bool get_property \(int prop, int &val\) const`](#)
Get integer property.
- [`bool get_property \(int prop, double &val\) const`](#)
Get real property.

Protected Attributes

- `opc_enum opc`
OPC state.
- `property_map opc_prop`
List of OPC properties.

8.43.1 Detailed Description

OPC list.

This class stores OPC properties.

Definition at line 100 of file ParseUtil.h.

The documentation for this class was generated from the following files:

- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.44 ParseUtil::optarg Class Reference

Optional arguments.

```
#include <ParseUtil.h>
```

Public Member Functions

- `optarg ()`
Default constructor.
- `optarg (const std::stringcase &arg)`
- `~optarg ()`
Destructor.
- `int parse (const std::stringcase &arg)`
- `int argc () const`
Returns the number of arguments.
- `const char *const * argv () const`
Returns the argument list.
- `const bool * argp () const`
Return the processed argument list.
- `bool * argp ()`
Return the processed argument list.
- `bool all_done () const`
Returns true if there all arguments are unprocessed.

Protected Member Functions

- `optarg` (const `optarg` &)
Disabled copy constructor.
- `optarg` & `operator=` (const `optarg` &)
Disabled assignment operator.
- `void setup` (int size)
Setup.

Protected Attributes

- `int mysize`
Size of allocated arrays.
- `int myargc`
Number of arguments.
- `char ** myargv`
Argument list.
- `bool * myargp`
Processed argument list.

8.44.1 Detailed Description

Optional arguments.

This class transforms a string into a standard program argument.

Definition at line 22 of file ParseUtil.h.

8.44.2 Constructor & Destructor Documentation

8.44.2.1 optarg()

```
ParseUtil::optarg::optarg (
    const std::string& arg) [inline], [explicit]
```

Constructor

Parameters

<code>arg</code>	Option argument string
------------------	------------------------

Definition at line 29 of file ParseUtil.h.

References `parse()`.

Here is the call graph for this function:



8.44.3 Member Function Documentation

8.44.3.1 parse()

```
int ParseUtil::optarg::parse (const std::string& arg)
```

Parse string argument

Parameters

<i>arg</i>	Option argument string
------------	------------------------

Returns

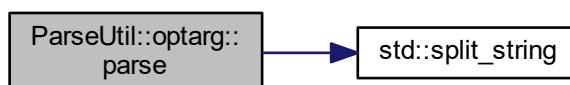
Number of processed arguments

Definition at line 14 of file ParseUtil.cpp.

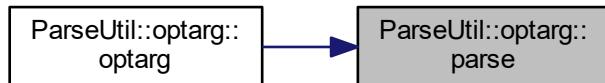
References std::split_string().

Referenced by optarg().

Here is the call graph for this function:



Here is the caller graph for this function:

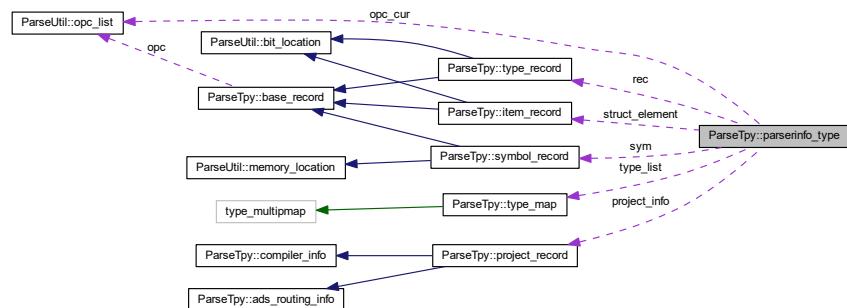


The documentation for this class was generated from the following files:

- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.45 ParseTpy::parserinfo_type Class Reference

Collaboration diagram for ParseTpy::parserinfo_type:



Public Member Functions

- [parserinfo_type \(project_record &p, symbol_list &s, type_map &t\)](#)
Constructor.
- [symbol_list & get_symbols \(\)](#)
Get symbol list.
- [type_map & get_types \(\)](#)
Get type list.
- [project_record & get_projectinfo \(\)](#)
Get project information.
- [void init \(\)](#)
Initialize temporary parser info.
- [type_enum get_type_description \(\) const](#)
Get type of parsed object.
- [bool verytop \(\)](#)
the very top of the xml tag hierarchy (not within any tag)
- [bool top \(\)](#)
the top of the xml tag hierarchy (within the PlcProjectInfo tag)

Public Attributes

- int `ignore`
ignore elements during parsing (with level)
- bool `projects`
parsing withing PlcProjectInfo tag
- int `routing`
parsing within Routing tag (1) or AdsInfo (2), Net ID (3), Port (4), Target name (5)
- int `compiler`
parsing within compiler tag (1) or compiler version (2), Twincat version (3), CPU family (4)
- int `types`
parsing within DataTypes tag (1) or DataType tag (2)
- int `symbols`
parsing within Symbols tag (1) or Symbol tag (2)
- `symbol_record sym`
temporary symbol information during parsing
- `type_record rec`
temporary type information during parsing
- int `name_parse`
- int `type_parse`
level indicator for type parsing
- `opc_list * opc_cur`
Pointer to current opc list (types only)
- `property_el opc_prop`
temporary opc element
- int `opc_parse`
level indicator for opc element
- `std::stringcase opc_data`
temporary data string for parsed opc data
- int `opc_cdata`
temporary cdata indicator
- int `igroup_parse`
level indicator for IGroup
- int `ioffset_parse`
level indicator for IOffset
- int `bitsize_parse`
level indicator for BitSize
- int `bitoffs_parse`
level indicator for BitOffs
- `std::stringcase data`
temporary data string for parsed igrup/ioffset/bitsize/bitoffs
- int `array_parse`
level indicator for array info parsing
- `std::stringcase array_data`
temporary data string for parsed array info
- `dimension array_bounds`
temporary array dimension element
- int `enum_parse`
level indicator for enum parsing
- `std::stringcase enum_data`
temporary data string for parsed enum data

- `enum_pair enum_element`
temporary enum element
- `std::stringcase enum_comment`
temporary enum comment
- `int struct_parse`
level indicator for struct parsing
- `item_record struct_element`
temporary structure element
- `int fb_parse`
level indicator for function block parsing

Protected Attributes

- `symbol_list * sym_list`
pointer to symbol list
- `type_map * type_list`
pointer to type list
- `project_record * project_info`
pointer to project info

8.45.1 Detailed Description

This structure keeps track of the parser information.

Definition at line 40 of file ParseTpy.cpp.

8.45.2 Member Data Documentation

8.45.2.1 name_parse

```
int ParseTpy::parserinfo_type::name_parse
```

level indicator for type name parsing level indicators in general: 0 - not encountered, 1 - parsed, 2 - currently processing, 3 - further sub tag parsing

Definition at line 96 of file ParseTpy.cpp.

The documentation for this class was generated from the following file:

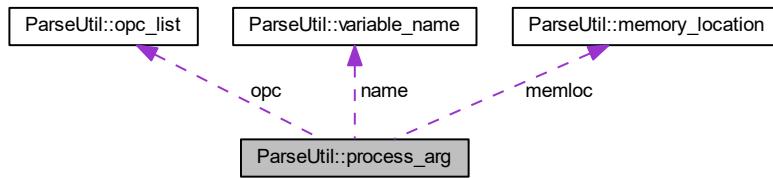
- `ParseTpy.cpp`

8.46 ParseUtil::process_arg Class Reference

Arguments for processing.

```
#include <ParseUtil.h>
```

Collaboration diagram for ParseUtil::process_arg:



Public Member Functions

- `process_arg (const memory_location &loc, const variable_name &vname, process_type_enum pt, const opc_list &o, const std::stringcase &tname, bool at)`
Get variable.
- `const variable_name & get_var () const`
Get name.
- `const std::stringcase & get_name () const`
Get alias.
- `const std::stringcase & get_alias () const`
Get type name.
- `const std::stringcase & get_process_type () const`
Get process type.
- `std::stringcase get_process_string () const`
Get process type string.
- `bool is_atomic () const`
Is atomic (or structured) type.
- `int get_igroup () const`
Get IGroup.
- `int get_ioffset () const`
Get IOOffset.
- `int get_bytesize () const`
Get BitSize.
- `std::stringcase get () const`
- `std::stringcase get_full () const`

Protected Attributes

- const `variable_name` & `name`
`name of type`
- const `std::stringcase` & `type_n`
`type definition`
- const `opc_list` & `opc`
`list of opc properties`
- const `memory_location` & `memloc`
`memory location`
- `process_type_enum ptype`
`Process Type.`
- bool `atomic`
`Atomic element.`

8.46.1 Detailed Description

Arguments for processing.

Argument which is passed to the name/tag processing function.

Definition at line 299 of file ParseUtil.h.

8.46.2 Constructor & Destructor Documentation

8.46.2.1 process_arg()

```
ParseUtil::process_arg::process_arg (
    const memory_location & loc,
    const variable_name & vname,
    process_type_enum pt,
    const opc_list & o,
    const std::stringcase & tname,
    bool at ) [inline]
```

Constructor

Parameters

<code>loc</code>	Memory location
<code>vname</code>	Variable name
<code>pt</code>	Process type
<code>o</code>	OPC list
<code>tname</code>	Type name
<code>at</code>	Atomic type

Definition at line 309 of file ParseUtil.h.

8.46.3 Member Function Documentation

8.46.3.1 get()

```
std::string ParseUtil::process_arg::get () const [inline]
```

Gets a string representation of a memory location

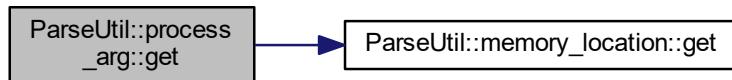
Returns

string with format "igroup/offset:size", empty on error

Definition at line 341 of file ParseUtil.h.

References ParseUtil::memory_location::get(), and memloc.

Here is the call graph for this function:



8.46.3.2 get_full()

```
std::string ParseUtil::process_arg::get_full () const
```

Gets a string representation of a PLC & memory location

Returns

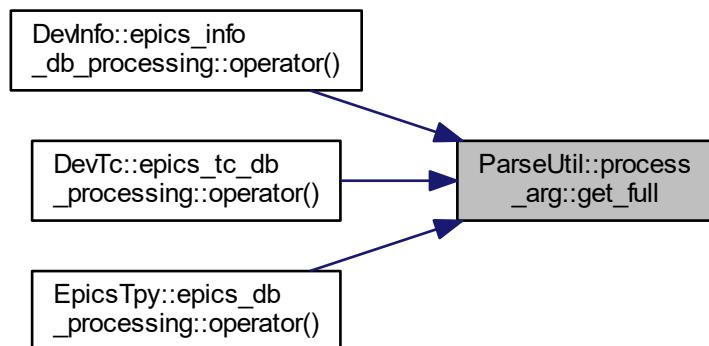
string with format "prefixigroup/offset:size", empty on error

Definition at line 267 of file ParseUtil.cpp.

References ParseUtil::OPC_PROP_PLCNAME.

Referenced by DevInfo::epics_info_db_processing::operator()(), DevTc::epics_tc_db_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

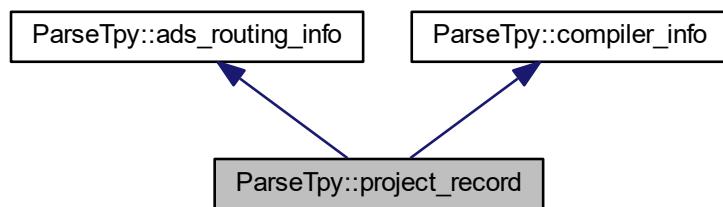
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.47 ParseTpy::project_record Class Reference

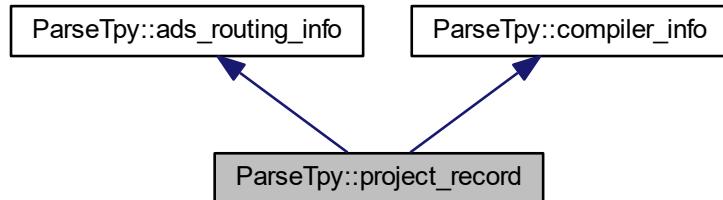
Project information.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::project_record:



Collaboration diagram for ParseTpy::project_record:



Public Member Functions

- [project_record \(\)](#)
Default constructor.

Additional Inherited Members

8.47.1 Detailed Description

Project information.

This is a base class for storing the project information

Definition at line 128 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.48 DevTc::register_devsup Class Reference

Device support registration.

```
#include <devTc.h>
```

Collaboration diagram for DevTc::register_devsup:



Public Types

- `typedef auto link_func(dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord) -> bool`
Type describing the link function.
- `typedef std::pair< std::regex, link_func & > test_pattern`
pair of pattern and link function
- `typedef std::vector< test_pattern > test_pattern_list`
list of pattern/link functions

Static Public Member Functions

- `static void add (const std::regex &rgx, link_func &func)`
Register a pattern/link function.
- `static bool linkRecord (const std::stringcase &inpout, dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`
linkRecord

Protected Member Functions

- `register_devsup ()`
Default constructor; adds a linkTcRecord entry.
- `register_devsup (const register_devsup &)`
Disabled copy constructor.
- `register_devsup & operator= (const register_devsup &)`
Disabled assignment operator.

Protected Attributes

- `test_pattern_list tp_list`
list of pattern and links

Static Protected Attributes

- `static register_devsup the_register_devsup`
the one global instance of the register class

8.48.1 Detailed Description

Device support registration.

This is a class for managing device support for multiple record types, such as TwinCAT/ADS and Info.

Definition at line 64 of file devTc.h.

The documentation for this class was generated from the following files:

- `devTc.h`
- `devTc.cpp`

8.49 DevInfo::register_info_devsup Class Reference

Epics interface class.

```
#include <devInfo.h>
```

Collaboration diagram for DevInfo::register_info_devsup:



Protected Member Functions

- [register_info_devsup \(\)](#)
Default constructor; adds a linkInfoRecord entry.

Static Protected Attributes

- static [register_info_devsup the_register_info_devsup](#)
Register info support.

8.49.1 Detailed Description

Epics interface class.

This is a class for an EPICS Interface

Definition at line 31 of file devInfo.h.

The documentation for this class was generated from the following files:

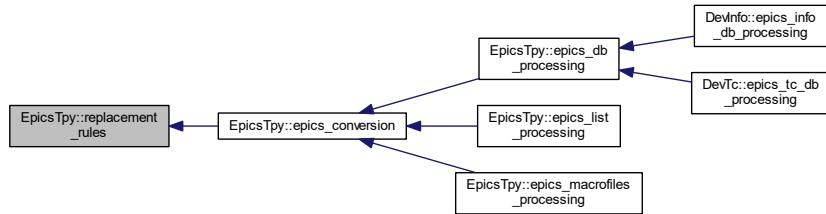
- [devInfo.h](#)
- [devInfo.cpp](#)

8.50 EpicsTpy::replacement_rules Class Reference

Replacement rules.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::replacement_rules:



Public Member Functions

- **`replacement_rules ()`**
Default constructor.
- **`replacement_rules (const replacement_table &t)`**
Constructor.
- **`void add_rule (const std::stringcase &var, const std::stringcase &val)`**
Add a rule.
- **`void set_rule_table (const replacement_table &t)`**
set table
- **`replacement_table & get_rule_table ()`**
get table
- **`const replacement_table & get_rule_table () const`**
get table
- **`std::stringcase apply_replacement_rules (const std::stringcase &s) const`**
replace
- **`bool HasRules () const`**
Has rules.

Static Public Attributes

- **`static const char *const prefix = "{$(`**
prefix for replacement rule: \$(
- **`static const char *const suffix = "}"`**
suffix for replacement rule:)

Protected Attributes

- **`replacement_table table`**
Replacement table.

8.50.1 Detailed Description

Replacement rules.

Epics channel conversion arguments Epics channels are generated from opc through a conversion rule

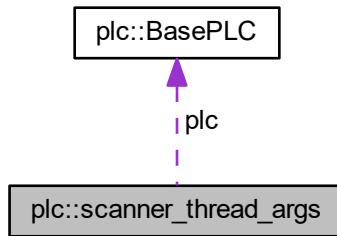
Definition at line 29 of file TpyToEpics.h.

The documentation for this class was generated from the following files:

- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.51 plc::scanner_thread_args Struct Reference

Collaboration diagram for plc::scanner_thread_args:



Public Attributes

- `plc::BasePLC * plc`
PLC this scanner operates on.
- `long scanperiod`
Period in ms of the scanner.
- `plc::BasePLC::scanner_func scanner`
Address of the scanner function to be used (read, write, or upddate)

8.51.1 Detailed Description

Structure for arguments sent to a scanner thread

Definition at line 645 of file plcBase.cpp.

The documentation for this struct was generated from the following file:

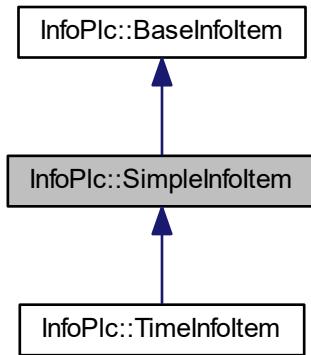
- [plcBase.cpp](#)

8.52 InfoPlc::SimpleInfoltem Class Reference

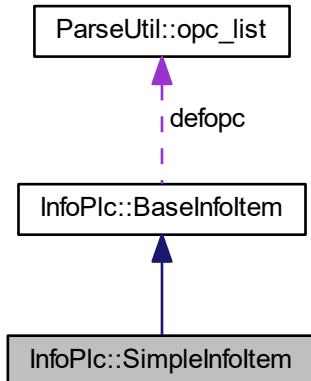
Simple info tracker.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::SimpleInfoltem:



Collaboration diagram for InfoPlc::SimpleInfoltem:



Public Member Functions

- `SimpleInfoltem ()`

Default constructor.

- **SimpleInfoItem** (const `ParseUtil::opc_list &opc`)
Initialize from OPC list.
- virtual bool **setup** (const `std::stringcase &tagname`, const `std::stringcase &plc_alias`, `plc::Interface *puser`) override
- virtual void **update** (double val) override
Recalculates last, min, max, avg, cnt based on a new value.
- virtual void **reset** () override
Reset.

Protected Attributes

- double **last**
Last.
- double **min**
Minimum.
- double **max**
Maximum.
- double **avg**
Average.
- double **cnt**
Count.

8.52.1 Detailed Description

Simple info tracker.

This is a class to keep simple statistics for a particular property

Definition at line 107 of file `infoPlc.h`.

8.52.2 Member Function Documentation

8.52.2.1 setup()

```
bool InfoPlc::SimpleInfoItem::setup (
    const std::stringcase & tagname,
    const std::stringcase & plc_alias,
    plc::Interface * puser ) [override], [virtual]
```

Setup

Parameters

<code>tagname</code>	Name of channels
<code>plc_alias</code>	PLC alias name
<code>puser</code>	Pointer ot PLC

SimpleInfoItem

Implements [InfoPlc::BaseInfoItem](#).

Definition at line 96 of file [infoPlc.cpp](#).

The documentation for this class was generated from the following files:

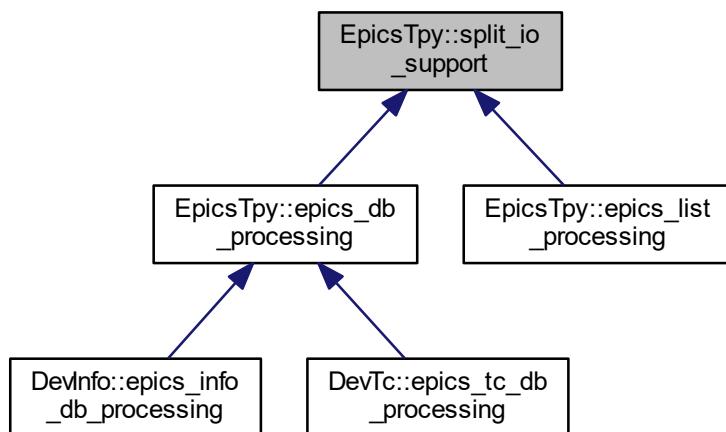
- [infoPlc.h](#)
- [infoPlc.cpp](#)

8.53 EpicsTpy::split_io_support Class Reference

Split IO support.

```
#include <TpyToEpics.h>
```

Inheritance diagram for EpicsTpy::split_io_support:



Public Member Functions

- [split_io_support \(\)](#)
Default constructor.
- [split_io_support \(const std::string& fname, bool split=false, int max=0\)](#)
Constructor.
- [split_io_support \(const std::string& fname, int argc, const char *const argv\[\], bool argp\[\] = 0\)](#)
- [~split_io_support \(\)](#)
Destructor.
- [split_io_support \(const split_io_support &\)](#)
- [split_io_support & operator= \(const split_io_support &\)](#)
- [bool operator! \(\) const](#)

- *Return error.*
 - bool **increment** (bool readonly)
 - void **flush** ()
 - Flush contents of output files.*
- FILE * **get_file** () const
 - Get output file.*
- const std::stringcase & **get_filename** () const
 - Get output filename.*
- bool **is_split** () const
 - Is output split?*
- int **get_max** () const
 - Maximum of channels per file.*
- int **get_processed_total** () const
 - Get number of processed channels.*
- int **get_processed_READONLY** () const
 - Get number of processed readonly channels.*
- int **get_processed_io** () const
 - Get number of processed input/output channels.*

Protected Member Functions

- void **set_filename** (const std::stringcase &fname)
 - Set output filename.*
- void **close** ()
 - Close files.*
- void **set_split** (bool split)
 - set split*
- void **set_max** (int max)
 - Set maximum of channels per file.*
- int **getopt** (int argc, const char *const argv[], bool argp[] = 0)

Protected Attributes

- bool **error**
 - Error.*
- std::stringcase **outfilename**
 - Output filename.*
- bool **split_io**
 - Split output into read only channels and input/output channels.*
- int **split_n**
 - Maximum number of channels per file; 0 indicates no limit.*
- FILE * **outf**
 - Output file.*
- FILE * **outf_in**
 - Output file for read only channels.*
- FILE * **outf_io**
 - Output file for input/output channels.*
- int **rec_num**
 - Current number of processed channels (records)*

- int `rec_num_in`
Current number of processed read only channels (records)
- int `rec_num_io`
Current number of processed input/output channels (records)
- int `file_num_in`
Current file number of processed read only channels (records)
- int `file_num_io`
Current file number of processed input/output channels (records)
- `std::stringcase file_num_in_s`
Contains the readonly file number in string format.
- `std::stringcase file_num_io_s`
Contains the input/output file number in string format.
- `std::stringcase file_in_s`
Contains the file extenstion for readonly files.
- `std::stringcase file_io_s`
Contains the file extenstion for input/output files.

8.53.1 Detailed Description

Split IO support.

Split file IO support Output can be split in multiple files if the number of channels exceeds the maximum specified for a file

Definition at line 204 of file TpyToEpics.h.

8.53.2 Constructor & Destructor Documentation

8.53.2.1 `split_io_support()` [1/2]

```
EpicsTpy::split_io_support::split_io_support (
    const std::stringcase & fname,
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Command line arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

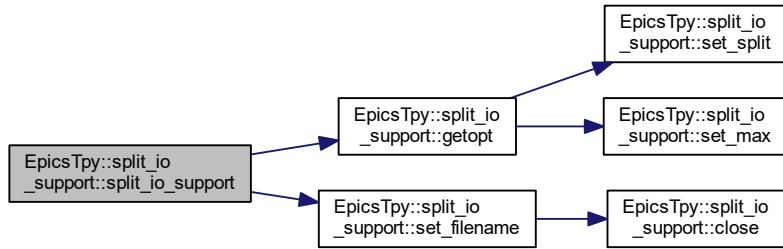
Parameters

<code>fname</code>	Filename for output
<code>argc</code>	Number of command line arguments
<code>argv</code>	List of command line arguments, same format as in main()
<code>argp</code>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 229 of file TpyToEpics.h.

References getopt(), and set_filename().

Here is the call graph for this function:



8.53.2.2 split_io_support() [2/2]

```
EpicsTpy::split_io_support::split_io_support (
    const split_io_support & iosup )
```

Copy constructor File pointers will be moved over and the original ones will become invalid.

Definition at line 240 of file TpyToEpics.cpp.

8.53.3 Member Function Documentation

8.53.3.1 getopt()

```
int EpicsTpy::split_io_support::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [protected]
```

Parse a command line The format is the same as the arguments passed to the main program `argv[0]` is program name and will be ignored The `argp` boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least `argc` long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

`/ysio`: Splits database into input only and input/output records
`/nsio`: Does not split database by record type (default)
`/sn 'num'`: Splits database or listing into files with no more than `num` records
`/sn 0`: Does not split database or listing into multiple files (default)

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. `getopt` will only override arguments that are specifically specified. It relies on the constructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

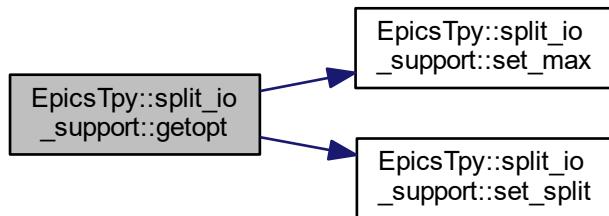
Number of arguments processed

Definition at line 276 of file TpyToEpics.cpp.

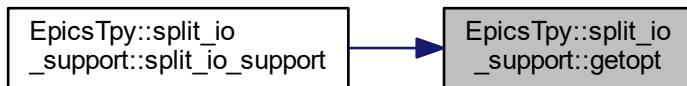
References [set_max\(\)](#), and [set_split\(\)](#).

Referenced by [split_io_support\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

**8.53.3.2 increment()**

```
bool EpicsTpy::split_io_support::increment (
    bool readonly )
```

Increase the channel number

Parameters

<i>readonly</i>	Indicates if channel is readonly
-----------------	----------------------------------

Returns

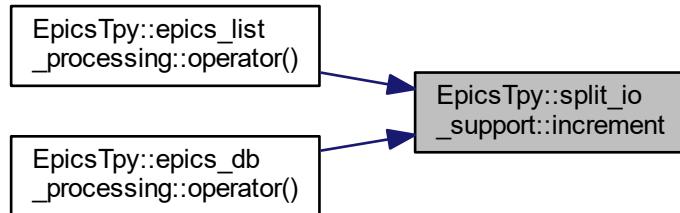
True if no error

Definition at line 406 of file TpyToEpics.cpp.

References error, file_in_s, file_io_s, file_num_in, file_num_in_s, file_num_io, file_num_io_s, outf, outf_in, outf_io, outfilename, rec_num, rec_num_in, rec_num_io, split_io, and split_n.

Referenced by EpicsTpy::epics_list_processing::operator()(), and EpicsTpy::epics_db_processing::operator()().

Here is the caller graph for this function:

**8.53.3.3 operator=()**

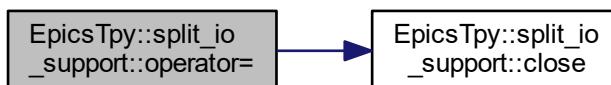
```
split_io_support & EpicsTpy::split_io_support::operator= (
    const split_io_support & iosup )
```

Assignment File pointers will be moved over and the original ones will become invalid.

Definition at line 249 of file TpyToEpics.cpp.

References close(), error, file_in_s, file_io_s, file_num_in, file_num_in_s, file_num_io, file_num_io_s, outf, outf_in, outf_io, outfilename, rec_num, rec_num_in, rec_num_io, split_io, and split_n.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

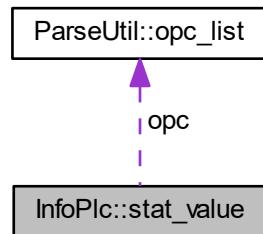
- [TpyToEpics.h](#)
- [TpyToEpics.cpp](#)

8.54 InfoPlc::stat_value Struct Reference

Statistic value.

```
#include <infoPlc.h>
```

Collaboration diagram for InfoPlc::stat_value:



Public Attributes

- [plc::BaseRecordPtr rec](#)
Record.
- [ParseUtil::opc_list opc](#)
OPC list.

8.54.1 Detailed Description

Statistic value.

This is a statistics value

Definition at line 45 of file infoPlc.h.

The documentation for this struct was generated from the following file:

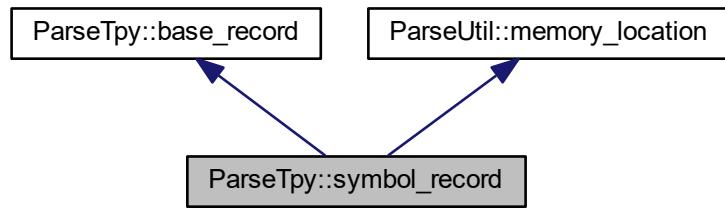
- [infoPlc.h](#)

8.55 ParseTpy::symbol_record Class Reference

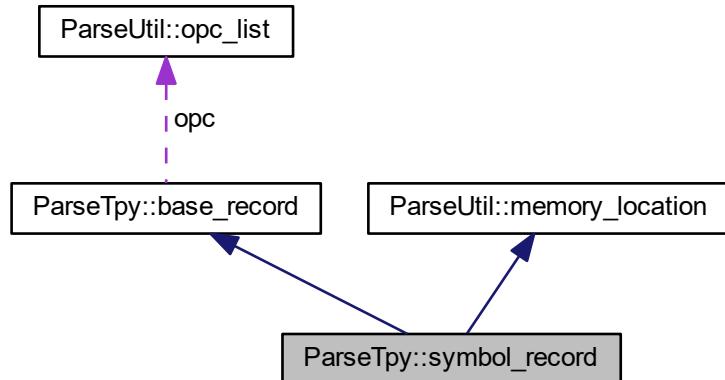
Symbol record.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::symbol_record:



Collaboration diagram for ParseTpy::symbol_record:



Public Member Functions

- [symbol_record \(\)](#)
Default constructor.
- [const memory_location & get_location \(\) const](#)
get memory location
- [memory_location & get_location \(\)](#)
get memory location

Additional Inherited Members

8.55.1 Detailed Description

Symbol record.

This structure holds a symbol record

Definition at line 345 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.56 syminfo_processing Class Reference

Public Member Functions

- [syminfo_processing](#) (FILE *outfile=0, bool atomic=true)
Constructor.
- [bool operator\(\)](#) (const [process_arg](#) &arg)
Process.

Protected Attributes

- [FILE * outf](#)
Ouptut file.
- [bool firstline](#)
Firstline?

8.56.1 Detailed Description

Symbol processing

Definition at line 16 of file ParseTpyInfo.cpp.

The documentation for this class was generated from the following file:

- [ParseTpyInfo.cpp](#)

8.57 plc::System Class Reference

[System.](#)

```
#include <plcBase.h>
```

Public Types

- `typedef std::recursive_mutex mutex_type`
Defines the mutex type.
- `typedef std::lock_guard< mutex_type > guard`
Defined the mutex guard type.

Public Member Functions

- `bool add (BasePLC *plc)`
- `bool add (BasePLCPtr plc)`
- `BasePLCPtr find (std::stringcase id)`
Finds a PLC by its name.
- `template<typename func >`
`void for_each (func &f)`
- `void printVals ()`
Print all PLC record values to the console.
- `void start ()`
Start scanning after ioc is running.
- `void stop ()`
Stop scanning when ioc is paused.
- `bool is_ioc_running () const`
get loc run state
- `void set_ioc_state (bool run)`
set loc run state

Static Public Member Functions

- `static System & get ()`
Return a reference to the global `System` variable.

Protected Attributes

- `mutex_type mux`
Mutex to synchronize access to this class.
- `BasePLCList PLCs`
Master list of all PLCs.
- `bool locRun`
IOC is running.

8.57.1 Detailed Description

`System`.

This is a class for managing multiple PLCs.

Definition at line 888 of file plcBase.h.

8.57.2 Member Function Documentation

8.57.2.1 add() [1/2]

```
bool plc::System::add (
    BasePLC * plc ) [inline]
```

Add a new PLC, PLC will be adopted

Parameters

<i>plc</i>	Pointer to plc
------------	----------------

Returns

true if successful

Definition at line 901 of file plcBase.h.

8.57.2.2 add() [2/2]

```
bool plc::System::add (
    BasePLCPtr plc )
```

Add a new PLC

Parameters

<i>plc</i>	Smart pointer to plc
------------	----------------------

Returns

true if successful

Definition at line 769 of file plcBase.cpp.

References mux, and PLCs.

8.57.2.3 for_each()

```
template<typename func >
void plc::System::for_each (
    func & f )
```

Iterate over all list elements This will yield good performance, but will lock the PLC for the entire processing time

Parameters

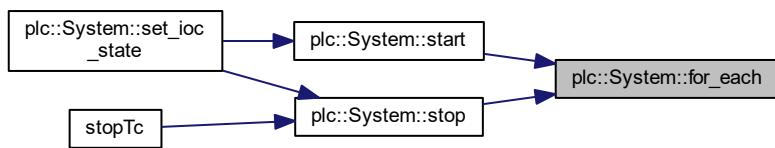
<code>f</code>	Function which takes <code>BaseRecord*</code> as the argument
----------------	---

Definition at line 216 of file `plcBaseTemplate.h`.

References `mux`, and `PLCs`.

Referenced by `start()`, and `stop()`.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

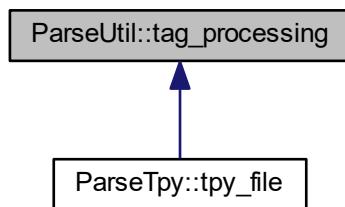
- `plcBase.h`
- `plcBase.cpp`
- `plcBaseTemplate.h`

8.58 ParseUtil::tag_processing Class Reference

Tag processing selection.

```
#include <ParseUtil.h>
```

Inheritance diagram for `ParseUtil::tag_processing`:



Public Member Functions

- `tag_processing ()`
Default constructor.
- `tag_processing (bool all, process_tag_enum proctags, bool nostring=false)`
- `tag_processing (int argc, const char *const argv[], bool argp[] = 0)`
- `int getopt (int argc, const char *const argv[], bool argp[] = 0)`
- `bool get_export_all () const`
Get the export rule.
- `void set_export_all (bool all)`
Set the export rule.
- `process_tag_enum get_process_tags () const`
Get the process rule.
- `void set_process_tags (process_tag_enum proctags)`
Set the process rule.
- `bool get_no_strings () const`
Get the string rule.
- `void set_no_strings (bool nostring)`
Set the string rule.

Protected Attributes

- `bool export_all`
Process all symbols regardless of opc publish setting.
- `process_tag_enum process_tags`
Process only atomic types.
- `bool no_string_tags`
Don't process strings.

8.58.1 Detailed Description

Tag processing selection.

Class to specify which symbols and tags/names to process

Definition at line 378 of file ParseUtil.h.

8.58.2 Constructor & Destructor Documentation

8.58.2.1 `tag_processing()` [1/2]

```
ParseUtil::tag_processing::tag_processing (
    bool all,
    process_tag_enum proctags,
    bool nostring = false ) [inline]
```

Constructor

Parameters

<i>all</i>	Process all tags
<i>proctags</i>	Process atomic and/or structured tags
<i>nostring</i>	Don't process string tags

Definition at line 388 of file ParseUtil.h.

8.58.2.2 tag_processing() [2/2]

```
ParseUtil::tag_processing::tag_processing (
    int argc,
    const char *const argv[],
    bool argp[] = 0 ) [inline]
```

Constructor Commaline arguments will override default parameters when specified The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored

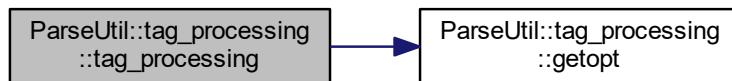
Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Definition at line 399 of file ParseUtil.h.

References getopt().

Here is the call graph for this function:

**8.58.3 Member Function Documentation**

8.58.3.1 getopt()

```
int ParseUtil::tag_processing::getopt (
    int argc,
    const char *const argv[],
    bool argp[] = 0 )
```

Parse a command line The format is the same as the arguments passed to the main program argv[0] is program name and will be ignored The argp boolean array can be used to pass in a list of already processed (and to be ignored) command line arguments. This list, if supplied, must be at least argc long. Upon return, newly processed arguments are also marked as processed in this list. The arguments are:

/ea: Export all variables regardless of OPC setting /eo: Only export variables which are marked as OPC export (default) /ns: No string variables are processed /ys: String variables are processes (default) /pa: Call process for all types (default) /ps: Call process for simple (atomic) types only /pc: Call process for complex (structure and array) types only

Command line arguments can use '-' instead of a '/'. Capitalization does not matter. getopt will only override arguments that are specifically specified. It relies on the contructors to provide the defaults.

Parameters

<i>argc</i>	Number of command line arguments
<i>argv</i>	List of command line arguments, same format as in main()
<i>argp</i>	Excluded/processed arguments (in/out), array length must be argc

Returns

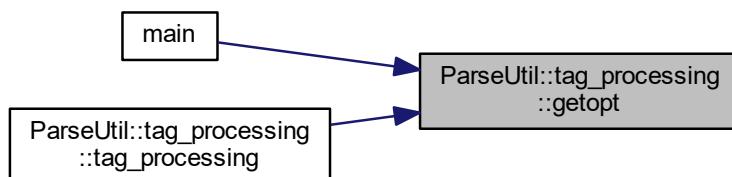
Number of arguments processed

Definition at line 283 of file ParseUtil.cpp.

References ParseUtil::process_all, ParseUtil::process_atomic, and ParseUtil::process_structured.

Referenced by [main\(\)](#), and [tag_processing\(\)](#).

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

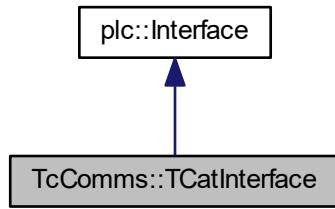
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

8.59 TcComms::TCatInterface Class Reference

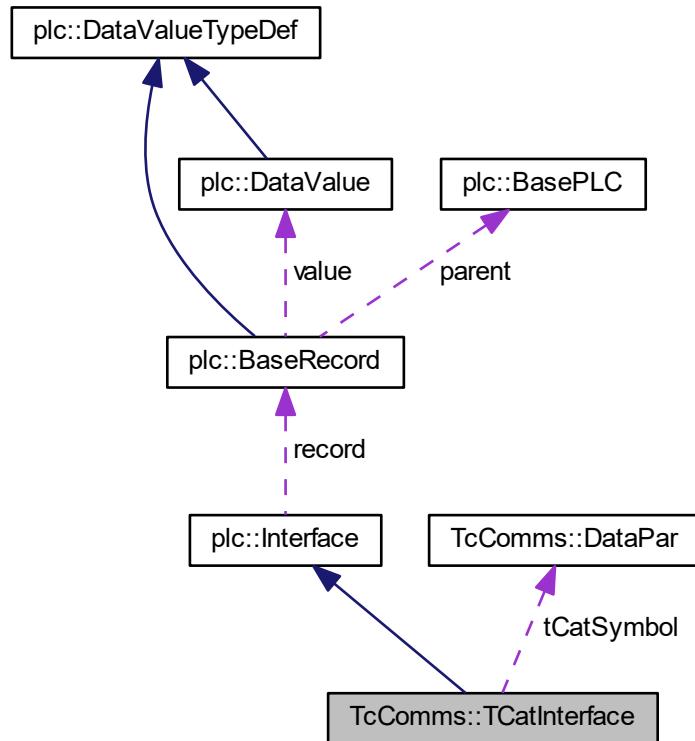
TCat interface class.

```
#include <tccomms.h>
```

Inheritance diagram for TcComms::TCatInterface:



Collaboration diagram for TcComms::TCatInterface:



Public Member Functions

- `TCatInterface (plc::BaseRecord &dval)`
Constructor.
- `TCatInterface (plc::BaseRecord &dval, const std::stringcase &name, unsigned long group, unsigned long offset, unsigned long length, const std::stringcase &type, bool isStruct, bool isEnum)`
- `~TCatInterface ()`
Deconstructor.
- `const std::stringcase get_tCatName ()`
Get name of TCat symbol.
- `void set_tCatName (std::stringcase name)`
Set name of TCat symbol.
- `const std::stringcase get_tCatype ()`
Get TCat data type.
- `void set_tCatype (std::stringcase type)`
Set TCat data type.
- `DataPar get_tCatsymbol ()`
Get structure containing index group, index offset, size.
- `unsigned long get_indexGroup () const`
Get index group.
- `void set_indexGroup (unsigned long group)`
Set index group.
- `unsigned long get_indexOffset () const`
Get index offset.
- `void set_indexOffset (unsigned long offset)`
Set index offset.
- `unsigned long get_size () const`
Get size in bytes of symbol.
- `void set_size (unsigned long nBytes)`
Set size in bytes of symbol.
- `size_t get_requestOffs () const`
Get offset into response buffer.
- `void set_requestOffs (size_t pVal)`
Set offset into response buffer.
- `TcPLC * get_parent ()`
Get parent PLC that owns this record.
- `const TcPLC * get_parent () const`
Get parent PLC that owns this record.
- `int get_requestNum ()`
Get the request group number this record is in.
- `void set_requestNum (int rNum)`
Set the request group number this record is in.
- `void printTCatVal (FILE *fp)`
- `virtual bool push () override`
Does nothing.
- `virtual bool pull () override`
Does nothing.

Protected Attributes

- `std::stringcase tCatName`
Name of TCat symbol.
- `std::stringcase tCatType`
Data type in TCat.
- `DataPar tCatSymbol`
Struct storing index group, index offset, and length of TC symbol.
- `int requestNum`
Which request group in the PLC.
- `size_t requestOffs`
Offset into response buffer.

8.59.1 Detailed Description

TCat interface class.

This is a class for a TCat interface

Definition at line 74 of file tcComms.h.

8.59.2 Constructor & Destructor Documentation

8.59.2.1 TCatInterface()

```
TcComms::TCatInterface::TCatInterface (
    plc::BaseRecord & dval,
    const std::stringcase & name,
    unsigned long group,
    unsigned long offset,
    unsigned long length,
    const std::stringcase & type,
    bool isStruct,
    bool isEnum )
```

Constructor

Parameters

<code>dval</code>	BaseRecord that this interface is part of
<code>name</code>	Name of TCat symbol
<code>group</code>	Index group of TCat symbol
<code>offset</code>	Index offset of TCat symbol
<code>length</code>	Size in bytes of TCat symbol
<code>type</code>	Name of TCat data type
<code>isStruct</code>	True = this symbol is a structure in TCat
<code>isEnum</code>	True = this symbol is an enum in TCat

Definition at line 87 of file tcComms.cpp.

References TcComms::DataPar::indexGroup, TcComms::DataPar::indexOffset, TcComms::DataPar::length, plc::←Interface::record, plc::BaseRecord::set_process(), tCatSymbol, and tCatType.

Here is the call graph for this function:



8.59.3 Member Function Documentation

8.59.3.1 printTCatVal()

```
void TcComms::TCatInterface::printTCatVal (
    FILE * fp )
```

Prints TCat symbol value and information

Parameters

<i>fp</i>	File to print symbol to
-----------	-------------------------

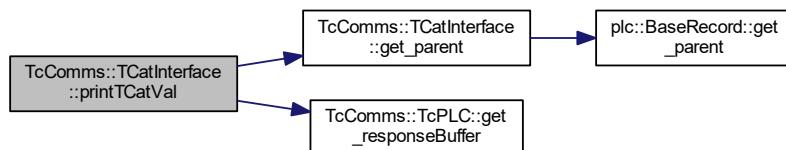
This is a function for printing the variable name and value of a record. Depending on the variable type, the readout from the ADS server is cast into the proper data type and printed to the output file *fp*.

Definition at line 122 of file tcComms.cpp.

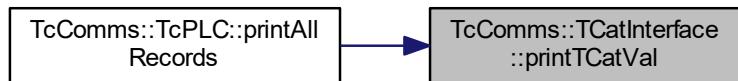
References get_parent(), TcComms::TcPLC::get_responseBuffer(), TcComms::DataPar::length, requestNum, requestOffs, tCatName, tCatSymbol, and tCatType.

Referenced by TcComms::TcPLC::printAllRecords().

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

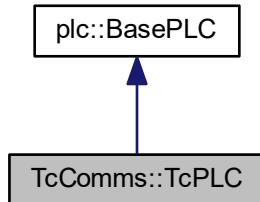
- [tcComms.h](#)
- [tcComms.cpp](#)

8.60 TcComms::TcPLC Class Reference

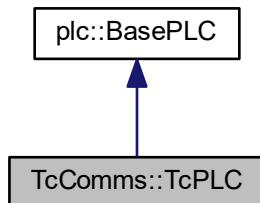
TwinCAT PLC.

```
#include <tcComms.h>
```

Inheritance diagram for TcComms::TcPLC:



Collaboration diagram for TcComms::TcPLC:



Public Types

- `typedef char buffer_type`
Buffer type.
- `typedef std::shared_ptr< buffer_type > buffer_ptr`
Smart pointer to buffer.

Public Member Functions

- `TcPLC (std::string tpyPath)`
Constructor.
- `~TcPLC ()`
Destructor.
- `bool is_valid_tpy ()`
Is typ still valid? Meaning, it hasn't changed.
- `AmsAddr get_addr ()`
Get AMS netID of TwinCAT system and port number for this PLC.
- `bool set_addr (std::string& netid, int port)`
- `long get_nReadPort ()`
Get read port number.
- `long get_nWritePort ()`
Get write port number.
- `int get_nRTS ()`
Get run-time system number.
- `void set_nRTS (int RTS)`
Set run-time system number.
- `void set_read_scanner_multiple (int mult)`
Set slowdown multiple for EPICS read.
- `ADSSTATE get_ads_state () const`
Get ADS state.
- `bool is_read_active () const`
Is read scanner active and successful.
- `virtual bool start ()`
Starts the appropriate scanners.
- `bool optimizeRequests ()`
- `buffer_ptr get_responseBuffer (size_t idx)`
Get pointer to the beginning of a read request response buffer.
- `virtual void printAllRecords ()`
Prints symbol information for entire list of symbols to console.

Protected Member Functions

- `virtual void read_scanner ()`
Makes read requests to ADS, makes PlcWrite on all data values.
- `virtual void write_scanner ()`
Collects records to be written to TCat, makes write request.
- `virtual void update_scanner ()`
Makes sure we don't have stale values.
- `void set_ads_state (ADSSTATE state)`
Set ADS state.

- void `setup_ads_notification ()`
Set up ADS status change notification.
- void `remove_ads_notification ()`
Remove ADS status change notification.
- long `openPort ()`
Opens a new ADS communication port.
- void `closePort (long nPort)`

Protected Attributes

- std::mutex `sync`
Mutex.
- AmsAddr `addr`
AMS netID of TwinCAT system and port number for this PLC.
- int `nRTS`
Run-time system number.
- std::string `pathTpy`
The path of the tpy file.
- time_t `timeTpy`
Modification time of file.
- std::atomic< bool > `checkTpy`
need to check modification time to make sure tpy file hasn't changed
- bool `validTpy`
tpy file is valid and hasn't changed
- int `nRequest`
Number of read request groups.
- std::vector< DataPar > `adsGroupReadRequestVector`
Vector of index group, index offset, size for read requests.
- std::vector< buffer_ptr > `adsResponseBufferVector`
Vector of buffers for each read request group.
- int `scanRateMultiple`
Slowdown multiple for EPICS read.
- int `cyclesLeft`
- int `update_workload`
Workload for update scanner.
- plc::BaseRecordPtr `update_last`
last updated record
- std::atomic< ADSSTATE > `ads_state`
ADS state.
- unsigned long `ads_handle`
ADS handle.
- std::atomic< bool > `ads_restart`
ADS restart.
- long `nReadPort`
Port number for ADS read connection.
- long `nWritePort`
Port number for ADS write connection.
- long `nNotificationPort`
Port number for ADS notification connection.
- bool `read_active`
read active and successful

Friends

- void __stdcall **ADScallback** (AmsAddr *, AdsNotificationHeader *, unsigned long)
Notification callback is a friend.

8.60.1 Detailed Description

TwinCAT PLC.

Class for a connection to a TwinCAT PLC This class is derived from a BasePLC object, and specializes in managing records that contain a plc interface for TCat. This class is initialized using a .tpy file, from which it will obtain the AMS address information for connecting with TCat through ADS.

Reading and writing from/to ADS will be managed by this class, with read requests being grouped by continuous memory region in TCat to optimize read scanning for speed. Write requests are made using an ADS sum request.

There is also an option to send a request to ADS to check the status of both the PLC device and also the ADS connection.

Definition at line 260 of file tcComms.h.

8.60.2 Member Function Documentation

8.60.2.1 closePort()

```
void TcComms::TcPLC::closePort (
    long nPort ) [protected]
```

Closes an ADS communication port

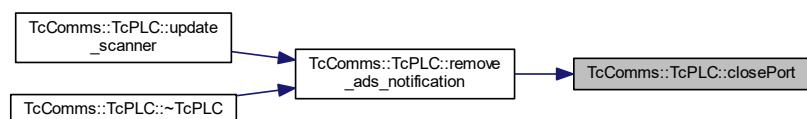
Parameters

<i>nPort</i>	Number of port to close
--------------	-------------------------

Definition at line 738 of file tcComms.cpp.

Referenced by `remove_ads_notification()`.

Here is the caller graph for this function:



8.60.2.2 optimizeRequests()

```
bool TcComms::TcPLC::optimizeRequests ( )
```

Sorts read channels into request groups. Will make a new request group for channels not in continuous memory region in TCat. Will create buffers of appropriate size for each read request, and let each TCat record know where in the read response buffer the data for that symbol is.

Returns

true if successful

Definition at line 427 of file tcComms.cpp.

8.60.2.3 set_addr()

```
bool TcComms::TcPLC::set_addr (
    std::stringcase netid,
    int port )
```

Set AMS address

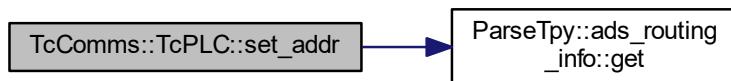
Returns

true if successful

Definition at line 320 of file tcComms.cpp.

References addr, and ParseTpy::ads_routing_info::get().

Here is the call graph for this function:



8.60.2.4 set_ads_state()

```
void TcComms::TcPLC::set_ads_state (
    ADSSTATE state ) [protected]
```

Set ADS state.

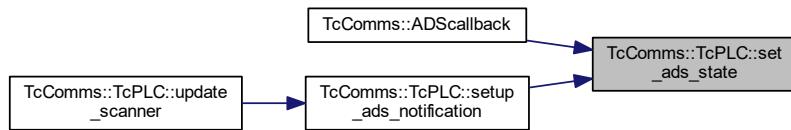
[TcPLC::set_ads_state](#)

Definition at line 566 of file tcComms.cpp.

References ads_state, checkTpy, and plc::BasePLC::name.

Referenced by TcComms::ADSCallback(), and setup_ads_notification().

Here is the caller graph for this function:



8.60.2.5 update_scanner()

```
void TcComms::TcPLC::update_scanner ( ) [protected], [virtual]
```

Makes sure we don't have stale values.

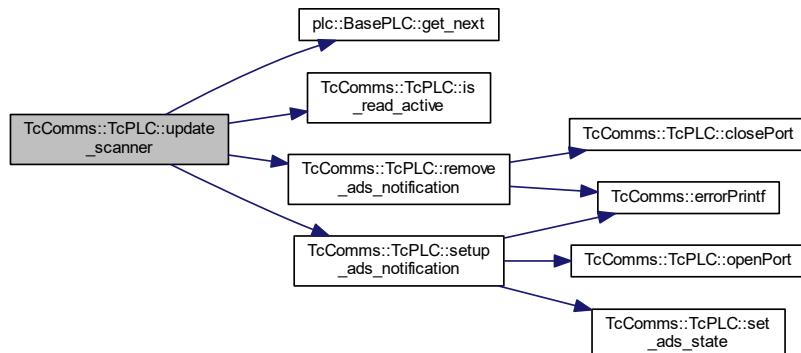
what!?

Reimplemented from [plc::BasePLC](#).

Definition at line 705 of file tcComms.cpp.

References ads_restart, plc::BasePLC::get_next(), is_read_active(), plc::BasePLC::mux, plc::BasePLC::name, plc::BasePLC::records, remove_ads_notification(), setup_ads_notification(), update_last, and update_workload.

Here is the call graph for this function:



8.60.3 Friends And Related Function Documentation

8.60.3.1 ADScallback

```
void __stdcall ADScallback (
    AmsAddr * pAddr,
    AdsNotificationHeader * pNotification,
    unsigned long plcId ) [friend]
```

Notification callback is a friend.

Callback for ADS state change

Definition at line 545 of file tcComms.cpp.

Referenced by setup_ads_notification().

8.60.4 Member Data Documentation

8.60.4.1 cyclesLeft

```
int TcComms::TcPLC::cyclesLeft [protected]
```

Cycles until EPICS read will be made Counts down from scanRateMultiple, resets at 0

Definition at line 364 of file tcComms.h.

Referenced by read_scanner().

The documentation for this class was generated from the following files:

- [tcComms.h](#)
- [tcComms.cpp](#)

8.61 TcComms::tcProcWrite Class Reference

TwinCAT process write requests.

```
#include <tcComms.h>
```

Public Member Functions

- `tcProcWrite` (const AmsAddr &a, long amsport, size_t mrec=1000)
Default constructor.
- `~tcProcWrite ()`
Destructor: will process the TCat writes.
- `tcProcWrite (tcProcWrite &&tp)`
Move constructor.
- `void operator() (plc::BaseRecord *prec)`
Process on record.
- `void * read_ptr (int sz)`
- `bool add (long igrp, long ioffs, long sz)`

Protected Member Functions

- `bool check_alloc (int extra=0)`
Checks if we have enough memory allocated.
- `void tcwrite ()`
writes the current header/data to TCat
- `tcProcWrite & operator= (tcProcWrite &&)`
Move operator.

Protected Attributes

- `AmsAddr addr`
AMS address.
- `long port`
Port to be used to write to TCat.
- `char * ptr`
Pointer to header to be written.
- `char * data`
Pointer to data to be written.
- `size_t maxrec`
Maximum number of individual requests.
- `size_t size`
Size of data.
- `size_t alloc`
Size of allocated header/data array.
- `size_t count`
Current number of individual requests.
- `std::vector< tcProcWrite > req`

8.61.1 Detailed Description

TwinCAT process write requests.

Class for collecting and processing write requests This class iterates through the entire record list on the PLC and collects those records whose data value has a dirty flag set on the plc side. These records are then sent as a group to ADS.

In order to not overload the ADS server, a maximum number of symbols per request is defined, and should not be > 2000.

Definition at line 182 of file tcComms.h.

8.61.2 Member Function Documentation

8.61.2.1 add()

```
bool TcComms::tcProcWrite::add (
    long igrup,
    long ioffs,
    long sz )
```

Add header info

Parameters

<i>igrup</i>	iGroup number for tc write
<i>ioffs</i>	iOffset number for tc write
<i>sz</i>	Size of data to be written

Returns

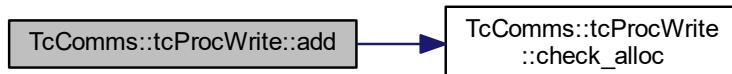
true if successful

Definition at line 247 of file tcComms.cpp.

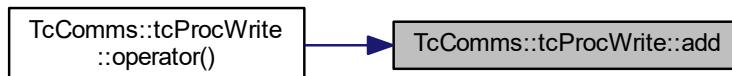
References check_alloc(), count, maxrec, ptr, and req.

Referenced by operator()().

Here is the call graph for this function:



Here is the caller graph for this function:



8.61.2.2 `read_ptr()`

```
void * TcComms::tcProcWrite::read_ptr (
    int sz )
```

Get a pointer to read the value in

Parameters

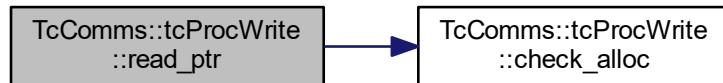
<code>sz</code>	Requested size
-----------------	----------------

Definition at line 235 of file tcComms.cpp.

References `check_alloc()`, `data`, and `size`.

Referenced by `operator()()`.

Here is the call graph for this function:



Here is the caller graph for this function:



8.61.3 Member Data Documentation

8.61.3.1 `req`

```
std::vector<tcProcWrite> TcComms::tcProcWrite::req [protected]
```

Queued TCat requests (each would have reached maxrec individual requests)

Definition at line 227 of file tcComms.h.

Referenced by `add()`, `operator=()`, and `~tcProcWrite()`.

The documentation for this class was generated from the following files:

- [tcComms.h](#)
- [tcComms.cpp](#)

8.62 DevTc::tcRegisterTolocShell Class Reference

Register TC commands.

```
#include <drvTc.h>
```

8.62.1 Detailed Description

Register TC commands.

Register TC commands to IOC shell This class registers the callback functions for the TC IOC commands

Definition at line 18 of file drvTc.h.

The documentation for this class was generated from the following files:

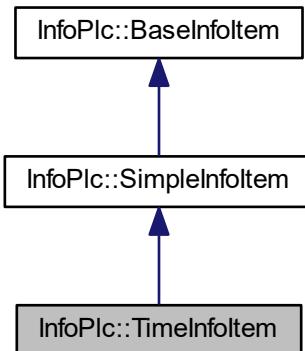
- [drvTc.h](#)
- [drvTc.cpp](#)

8.63 InfoPlc::TimeInfoltem Class Reference

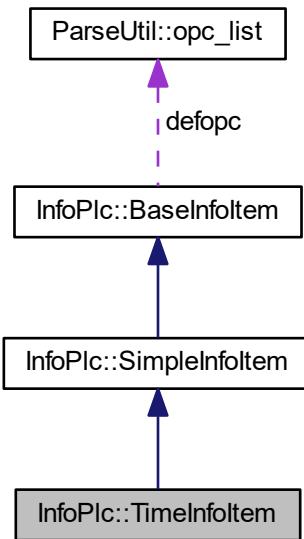
Timing tracker.

```
#include <infoPlc.h>
```

Inheritance diagram for InfoPlc::TimeInfoltem:



Collaboration diagram for InfoPlc::TimeInfoltem:



Public Member Functions

- `TimeInfoltem ()`
Default constructor.
- `~TimeInfoltem ()`
Destructor.
- `void start ()`
Start stopwatch.
- `void stop ()`
Stop stopwatch.

Protected Attributes

- `clock_t begin`
Start time.
- `clock_t end`
End time.
- `double elapsed`
Elapsed time.

8.63.1 Detailed Description

Timing tracker.

This is a class for timing a process

Definition at line 201 of file infoPlc.h.

8.63.2 Member Function Documentation

8.63.2.1 start()

```
void InfoPlc::TimeInfoItem::start ( )
```

Start stopwatch.

[TimeInfoItem](#)

Definition at line 143 of file infoPlc.cpp.

The documentation for this class was generated from the following files:

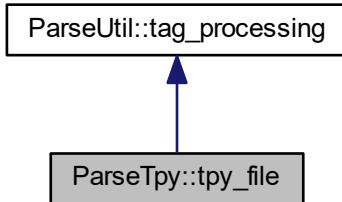
- [infoPlc.h](#)
- [infoPlc.cpp](#)

8.64 ParseTpy::tpy_file Class Reference

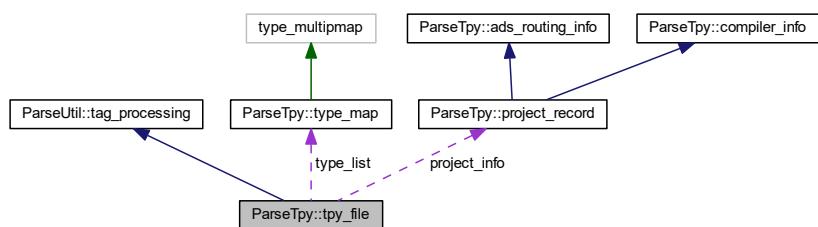
Tpy file parsing.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::tpy_file:



Collaboration diagram for ParseTpy::tpy_file:



Public Member Functions

- `tpy_file ()`
`Default constructor.`
- `tpy_file (FILE *inp)`
`Constructor.`
- `bool parse (FILE *inp)`
`Parse a file.`
- `bool parse (const char *p, int len)`
`Parse a memory region.`
- `const symbol_list & get_symbols () const`
`Return list of symbols.`
- `const type_map & get_types () const`
`Return list of types.`
- `const project_record & get_project_info () const`
`Return project information.`
- template<class Function >
`int process_symbols (Function &process, const std::stringcase &prefix=std::stringcase()) const`
`Process the type tree of a symbol.`
- template<class Function >
`int process_type_tree (const symbol_record &symbol, Function &process, const std::stringcase &prefix=std::stringcase()) const`
`Process the type tree of a symbol.`
- template<class Function >
`int process_type_tree (const type_record &typ, ParseUtil::opc_list defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname=ParseUtil::variable_name(), int level=0) const`
`Process the type tree of a type.`
- template<class Function >
`int process_type_tree (const std::stringcase &typ, unsigned int id, const ParseUtil::opc_list &defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname=ParseUtil::variable_name(), int level=0) const`
`Process the type tree of a type.`

Protected Member Functions

- `void parse_finish ()`
`finish up the parsing`
- template<class Function >
`int process_array (const type_record &typ, dimensions dim, const ParseUtil::opc_list &defopc, const ParseUtil::memory_location &loc, Function &process, const ParseUtil::variable_name &varname, int level) const`
`Process the type tree of a type.`

Protected Attributes

- `project_record project_info`
`Project information.`
- `symbol_list sym_list`
`List of symbols.`
- `type_map type_list`
`List of types.`

8.64.1 Detailed Description

Tpy file parsing.

This class holds the structure of a tpy file

Definition at line 370 of file ParseTpy.h.

8.64.2 Member Function Documentation

8.64.2.1 parse_finish()

```
void ParseTpy::tpy_file::parse_finish ( ) [protected]
```

finish up the parsing

This function is called at the end of parsing. Here we set the TC server name in the OPC variables for each symbol

Definition at line 480 of file ParseTpy.cpp.

References ParseUtil::OPC_PROP_PLCNAME.

8.64.2.2 process_array()

```
template<class Function >
int ParseTpy::tpy_file::process_array (
    const type_record & typ,
    dimensions dim,
    const ParseUtil::opc_list & defopc,
    const ParseUtil::memory_location & loc,
    Function & process,
    const ParseUtil::variable_name & varname,
    int level ) const [protected]
```

Process the type tree of a type.

Resolves the type information for an array. Calls the process function for each index with an argument of type process_arg.

Parameters

<i>typ</i>	Name of type to resolve
<i>dim</i>	Dimensions of the array
<i>defopc</i>	Default list of OPC parameters
<i>loc</i>	Memory location of variable
<i>process</i>	Function class
<i>varname</i>	Name of variable of the specified type (default is "")
<i>level</i>	Recursive level (stops when reaching 100, default 0)

Returns

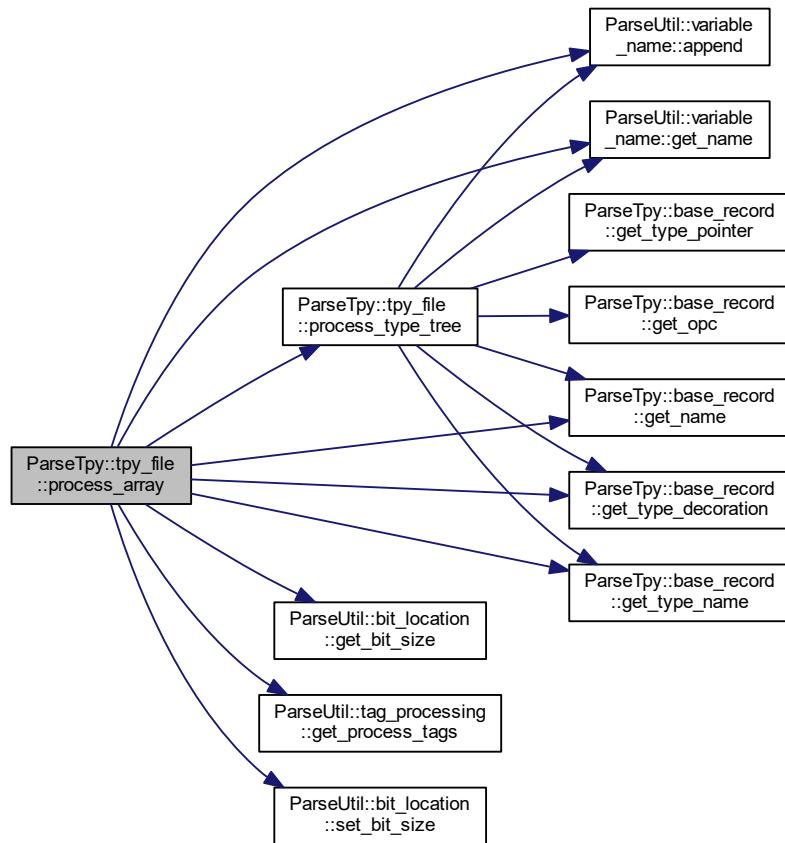
Number of processes variables

Definition at line 210 of file ParseTpyTemplate.h.

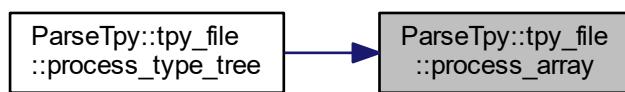
References ParseUtil::variable_name::append(), ParseUtil::bit_location::get_bit_size(), ParseUtil::variable_name::get_name(), ParseTpy::base_record::get_name(), ParseUtil::tag_processing::get_process_tags(), ParseTpy::base_record::get_type_decoration(), ParseTpy::base_record::get_type_name(), ParseUtil::process_all, ParseUtil::process_structured, process_type_tree(), ParseUtil::pt_binary, and ParseUtil::bit_location::set_bit_size().

Referenced by process_type_tree().

Here is the call graph for this function:



Here is the caller graph for this function:



8.64.2.3 process_symbols()

```
template<class Function >
int ParseTpy::tpy_file::process_symbols (
    Function & process,
    const std::stringcase & prefix = std::stringcase() ) const
```

Process the type tree of a symbol.

Iterates over the symbol list and processes all specified tags.

Parameters

<i>process</i>	Function class
<i>prefix</i>	Prefix which is added to all variable names

Returns

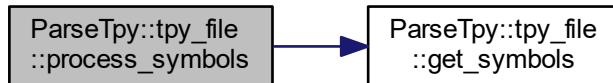
Number of processes variables

Definition at line 9 of file ParseTpyTemplate.h.

References get_symbols().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



8.64.2.4 process_type_tree() [1/3]

```
template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const symbol_record & symbol,
    Function & process,
    const std::stringcase & prefix = std::stringcase() ) const
```

Process the type tree of a symbol.

Starts with a symbol and resolves the type information recursively until an atomic type (like INT) is found. Then, calls the process function with an argument of type process_arg. The function must return true if successful and false otherwise.

Parameters

<i>symbol</i>	Symbol to resolve
<i>process</i>	Function class
<i>prefix</i>	Prefix which is added to all variable names

Returns

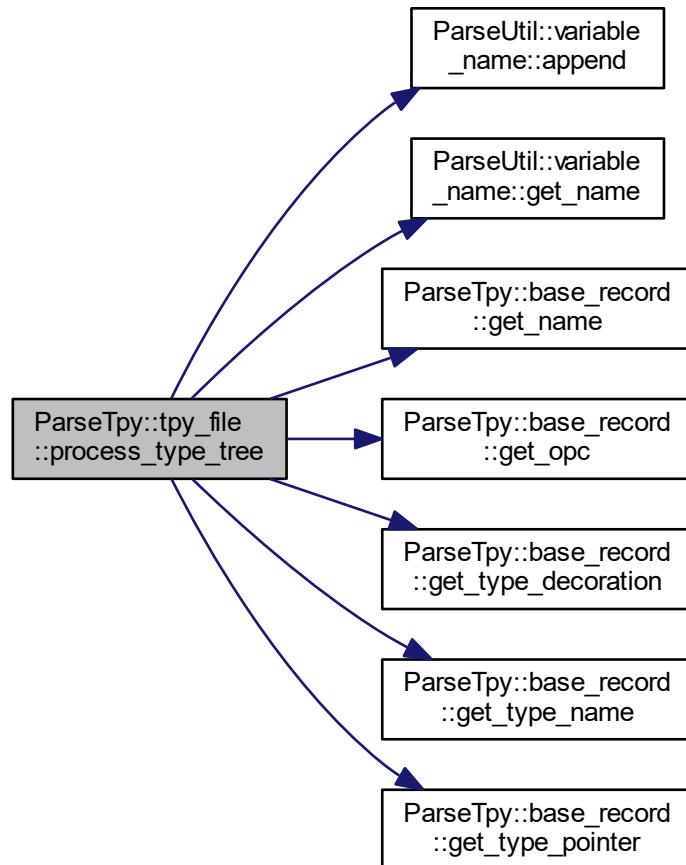
Number of processes variables

Definition at line 27 of file ParseTpyTemplate.h.

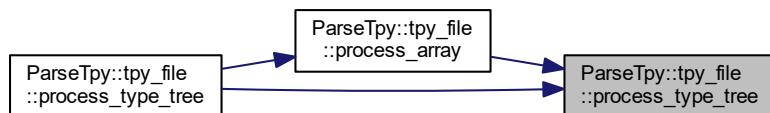
References ParseUtil::variable_name::append(), ParseUtil::variable_name::get_name(), ParseTpy::base_record::get_name(), ParseTpy::base_record::get_opc(), ParseTpy::base_record::get_type_decoration(), ParseTpy::base_record::get_type_name(), and ParseTpy::base_record::get_type_pointer().

Referenced by process_array(), and process_type_tree().

Here is the call graph for this function:



Here is the caller graph for this function:



8.64.2.5 process_type_tree() [2/3]

```
template<class Function >
int ParseTpy::tpy_file::process_type_tree (
```

```

const type_record & typ,
ParseUtil::opc_list defopc,
const ParseUtil::memory_location & loc,
Function & process,
const ParseUtil::variable_name & varname = ParseUtil::variable_name(),
int level = 0 ) const

```

Process the type tree of a type.

Starts with a type and resolves the type information recursively until an atomic type (like BOOL) is found. Then, calls the process function with an argument of type process_arg.

Parameters

<i>typ</i>	Type to resolve
<i>defopc</i>	Default list of OPC parameters
<i>loc</i>	Memory location of variable
<i>process</i>	Function class
<i>varname</i>	Name of variable of the specified type (default is "")
<i>level</i>	Recursive level (stops when reaching 100, default 0)

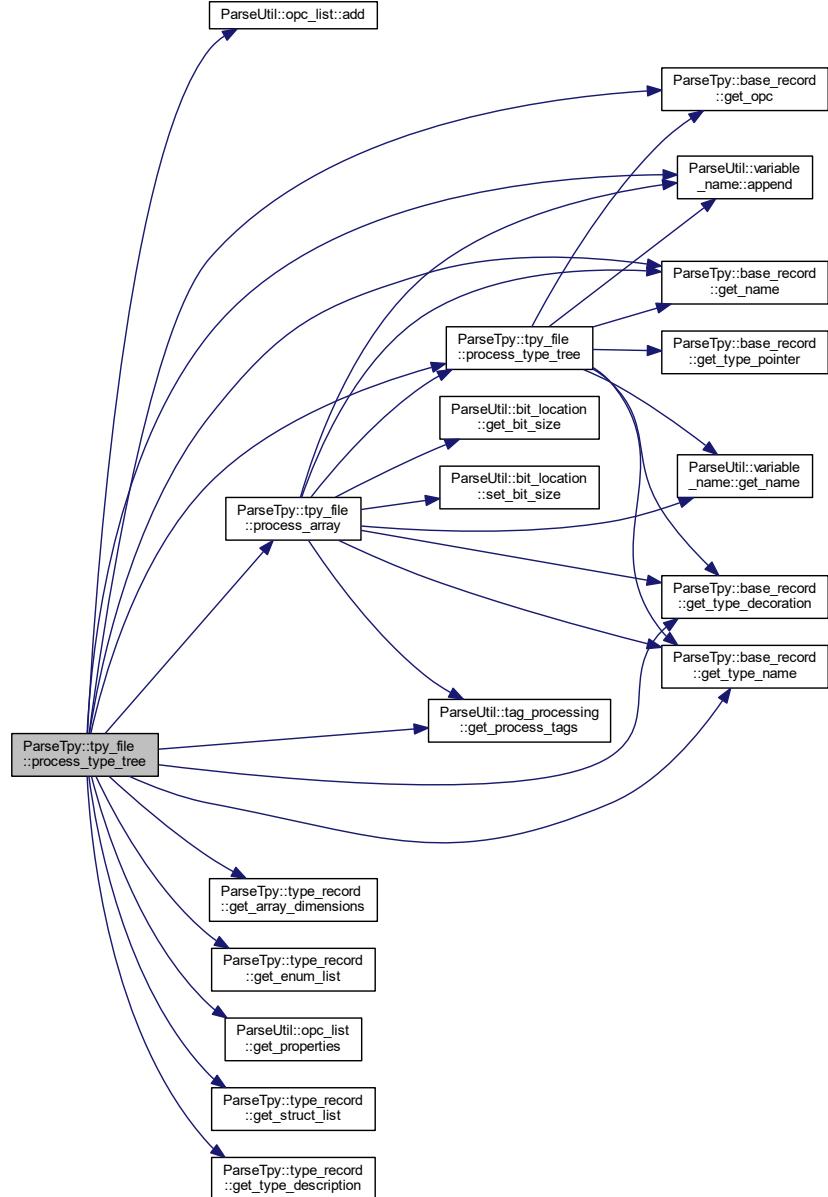
Returns

Number of processes variables

Definition at line 49 of file ParseTpyTemplate.h.

References ParseUtil::opc_list::add(), ParseUtil::variable_name::append(), ParseTpy::arraytype, ParseTpy::enumtype, ParseTpy::functionblock, ParseTpy::type_record::get_array_dimensions(), ParseTpy::type_record::get_enum_list(), ParseTpy::base_record::get_name(), ParseTpy::base_record::get_opc(), ParseUtil::tag_processing::get_process_tags(), ParseUtil::opc_list::get_properties(), ParseTpy::type_record::get_struct_list(), ParseTpy::base_record::get_type_decoration(), ParseTpy::type_record::get_type_description(), ParseTpy::base_record::get_type_name(), ParseUtil::process_all, process_array(), ParseUtil::process_atomic, ParseUtil::process_structured, process_type_tree(), ParseUtil::pt_binary, ParseUtil::pt_enum, ParseUtil::pt_int, ParseTpy::simple, and ParseTpy::structtype.

Here is the call graph for this function:



8.64.2.6 process_type_tree() [3/3]

```

template<class Function >
int ParseTpy::tpy_file::process_type_tree (
    const std::string& typ,
    unsigned int id,
    const ParseUtil::opc_list & defopc,
    const ParseUtil::memory_location & loc,

```

```

Function & process,
const ParseUtil::variable_name & varname = ParseUtil::variable_name(),
int level = 0 ) const

```

Process the type tree of a type.

Starts with a type and resolves the type information recursively until an atomic type (like STRING) is found. Then, calls the process function with an argument of type process_arg.

Parameters

<i>typ</i>	Name of type to resolve
<i>id</i>	Decoration or unique ID of type
<i>defopc</i>	Default list of OPC parameters
<i>loc</i>	Memory location of variable
<i>process</i>	Function class
<i>varname</i>	Name of variable of the specified type (default is "")
<i>level</i>	Recursive level (stops when reaching 100, default 0)

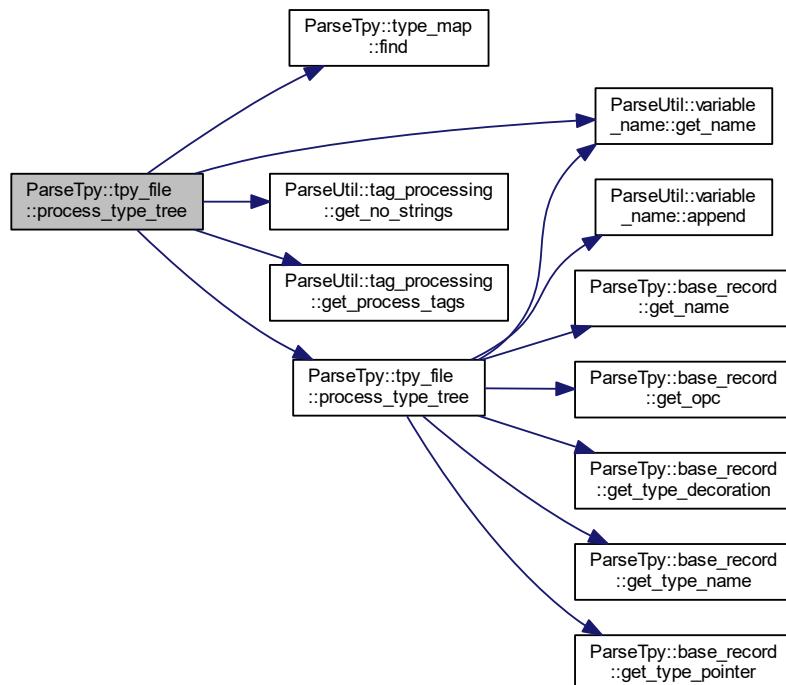
Returns

Number of processes variables

Definition at line 157 of file ParseTpyTemplate.h.

References ParseTpy::type_map::find(), ParseUtil::variable_name::get_name(), ParseUtil::tag_processing::get_no_strings(), ParseUtil::tag_processing::get_process_tags(), ParseUtil::process_all, ParseUtil::process_atomic, process_type_tree(), ParseUtil::pt_bool, ParseUtil::pt_int, ParseUtil::pt_real, ParseUtil::pt_string, and type_list.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)
- [ParseTpyTemplate.h](#)

8.65 DevTc::epics_record_traits< RecType >::traits_type Struct Reference

Epics record type.

```
#include <devTc.h>
```

Public Attributes

- double **val**
Value.

8.65.1 Detailed Description

```
template<epics_record_enum RecType>
struct DevTc::epics_record_traits< RecType >::traits_type
```

Epics record type.

Definition at line 262 of file devTc.h.

The documentation for this struct was generated from the following file:

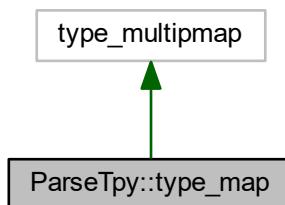
- [devTc.h](#)

8.66 ParseTpy::type_map Class Reference

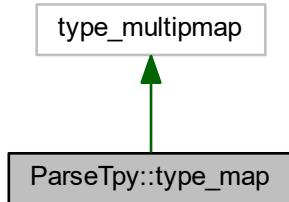
Type dictionary.

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::type_map:



Collaboration diagram for ParseTpy::type_map:



Public Member Functions

- [type_map \(\)](#)
Constructor.
- void [insert \(value_type val\)](#)
insert a new element
- const value_type::second_type * [find \(value_type::first_type id, const std::string& typn\) const](#)
find an element

8.66.1 Detailed Description

Type dictionary.

This is a map of type records, index is type number as defined in tpy

Definition at line 321 of file ParseTpy.h.

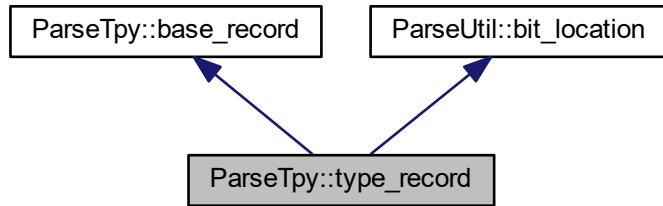
The documentation for this class was generated from the following files:

- [ParseTpy.h](#)
- [ParseTpy.cpp](#)

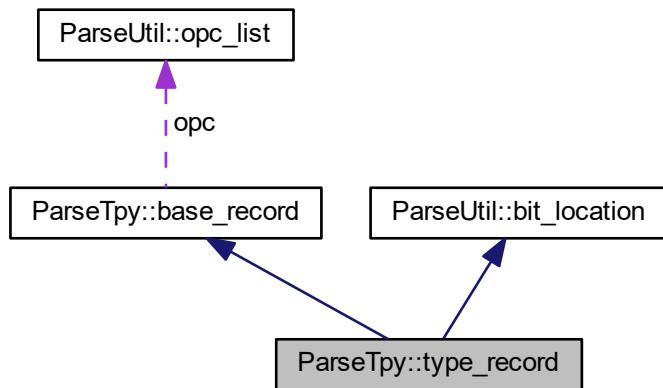
8.67 ParseTpy::type_record Class Reference

```
#include <ParseTpy.h>
```

Inheritance diagram for ParseTpy::type_record:



Collaboration diagram for ParseTpy::type_record:



Public Member Functions

- `type_record ()`
Default constructor.
- `type_enum get_type_description () const`
get the data type
- `void set_type_description (type_enum desc)`
Set decoration (type ID)
- `unsigned int get_name_decoration () const`
Get decoration (type ID)
- `void set_name_decoration (unsigned int id)`
Set decoration (type ID)
- `const dimensions & get_array_dimensions () const`
Get array dimensions.

- [dimensions & get_array_dimensions \(\)](#)
Get array dimensions.
- [const enum_map & get_enum_list \(\) const](#)
Get enumerated list.
- [enum_map & get_enum_list \(\)](#)
Get enumerated list.
- [const item_list & get_struct_list \(\) const](#)
Get structure list.
- [item_list & get_struct_list \(\)](#)
Get structure list.

Protected Attributes

- [type_enum type_desc](#)
Type description.
- [unsigned int name_decoration](#)
decoration (type ID) of type name
- [dimensions array_list](#)
table of dimensions
- [enum_map enum_list](#)
map of enum id and name
- [item_list struct_subitems](#)
list of structure elements

8.67.1 Detailed Description

This structure holds a type record

Definition at line 272 of file ParseTpy.h.

The documentation for this class was generated from the following file:

- [ParseTpy.h](#)

8.68 ParseUtil::variable_name Class Reference

Variable name.

```
#include <ParseUtil.h>
```

Public Member Functions

- [variable_name \(\)](#)
Default constructor.
- [variable_name \(const std::stringcase &n\)](#)
Constructor.
- [variable_name \(const std::stringcase &n, const std::stringcase &a\)](#)
Constructor.
- [const std::stringcase & get_name \(\) const](#)
Get name.
- [const std::stringcase & get_alias \(\) const](#)
Get alias.
- [void set \(const std::stringcase &n\)](#)
Set name & alias.
- [void set \(const std::stringcase &n, const std::stringcase &a\)](#)
Set name & alias.
- [void append \(const std::stringcase &n, const std::stringcase &sep="."\)](#)
Append.
- [void append \(const std::stringcase &n, const opc_list &opc, const std::stringcase &sep="."\)](#)
Append.

Protected Attributes

- [std::stringcase name](#)
variable name
- [std::stringcase alias](#)
alias name

8.68.1 Detailed Description

Variable name.

This is a class for storing a variable name and an alias

Definition at line 150 of file ParseUtil.h.

The documentation for this class was generated from the following files:

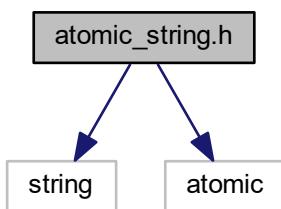
- [ParseUtil.h](#)
- [ParseUtil.cpp](#)

Chapter 9

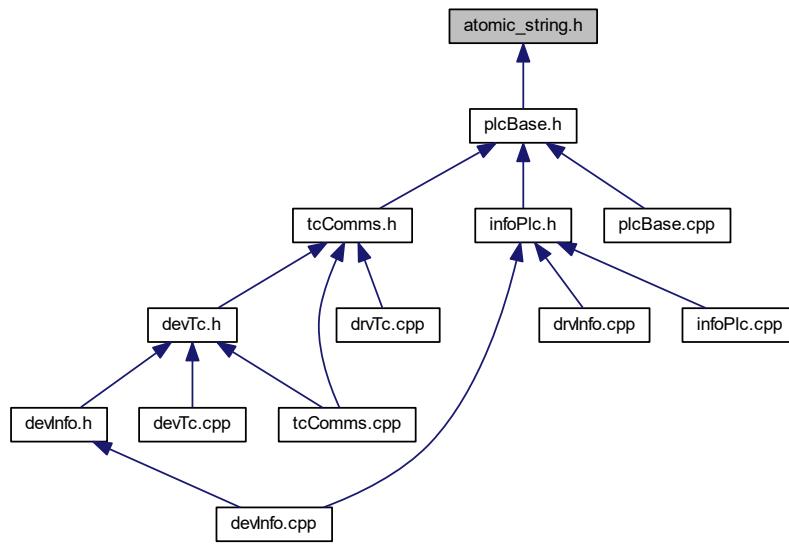
File Documentation

9.1 atomic_string.h File Reference

```
#include <string>
#include <atomic>
Include dependency graph for atomic_string.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `std::atomic_string< stringT >`
atomic strings
- class `std::atomic< string >`
atomic<string>
- class `std::atomic< wstring >`
atomic<wstring>

9.1.1 Detailed Description

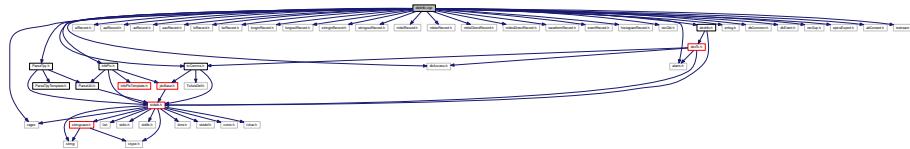
Header which includes a class for implementing an atomic specialization for string.

9.2 devInfo.cpp File Reference

```

#include "devInfo.h"
#include "ParseTpy.h"
#include "infoPlc.h"
#include "errlog.h"
#include "dbAccess.h"
#include "dbCommon.h"
#include "dbEvent.h"
#include "recSup.h"
#include "epicsExport.h"
#include "aitConvert.h"
#include <regex>
  
```

```
#include <iostream>
Include dependency graph for devInfo.cpp:
```



Namespaces

- [DevInfo](#)
Namespace for info device support.

Functions

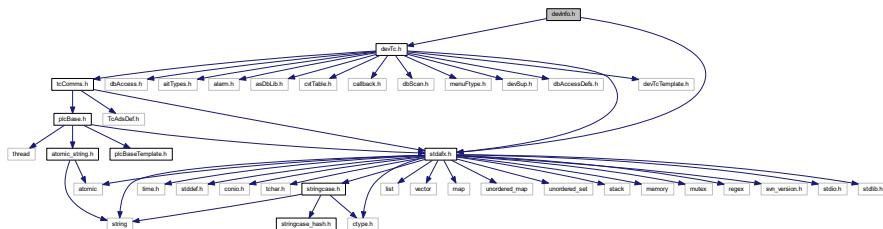
- bool [DevInfo::linkInfoRecord](#) (dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)
Link Information Record.
- [epicsExportAddress](#) (dset, aival_record_info_dset)
Record processing entry for ao.
- [epicsExportAddress](#) (dset, bival_record_info_dset)
Record processing entry for bi.
- [epicsExportAddress](#) (dset, longinval_record_info_dset)
Record processing entry for longin.
- [epicsExportAddress](#) (dset, mbbival_record_info_dset)
Record processing entry for mbbi.
- [epicsExportAddress](#) (dset, stringinval_record_info_dset)
Record processing entry for stringin.
- [epicsExportAddress](#) (dset, aoaval_record_info_dset)
Record processing entry for ao.
- [epicsExportAddress](#) (dset, boaval_record_info_dset)
Record processing entry for bo.
- [epicsExportAddress](#) (dset, longoutval_record_info_dset)
Record processing entry for longout.
- [epicsExportAddress](#) (dset, mbboaval_record_info_dset)
Record processing entry for mbbo.
- [epicsExportAddress](#) (dset, stringoutval_record_info_dset)
Record processing entry for stringout.

9.2.1 Detailed Description

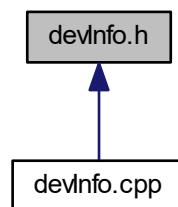
Methods for Info device support.

9.3 devInfo.h File Reference

```
#include "stdafx.h"
#include "devTc.h"
Include dependency graph for devInfo.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class DevInfo::register_info_devsup
Epics interface class.

Namespaces

- **DevTc**
Namespace for info device support.
 - **DevInfo**
Namespace for info device support.

Functions

- bool DevInfo::linkInfoRecord (dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)
Link Information Record.
 - const std::regex DevInfo::info_regex ("((info):://((\\b[0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?)+:([8-9][0-9]))|([\\d{1,9}])/([\\d{1,9}]):([\\d{1,9}])")
Regex for identifying TwinCAT records.

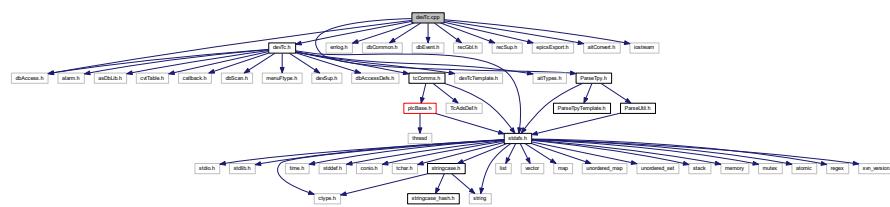
9.3.1 Detailed Description

Header which includes classes for Info device support.

9.4 devTc.cpp File Reference

```
#include "devTc.h"
#include "ParseTpy.h"
#include "errlog.h"
#include "dbAccess.h"
#include "dbCommon.h"
#include "dbEvent.h"
#include "recGbl.h"
#include "recSup.h"
#include "epicsExport.h"
#include "aitConvert.h"
#include <iostream>
```

Include dependency graph for devTc.cpp:



Namespaces

- `DevTc`

Namespace for info device support.

Functions

- `bool DevTc::linkRecord (std::string name, dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`
Link Record.
- `bool DevTc::linkTcRecord (dbCommon *pEpicsRecord, plc::BaseRecordPtr &pRecord)`
Link TwinCat Record.
- `epicsExportAddress (dset, aival_record_tc_dset)`
Record processing entry for ai.
- `epicsExportAddress (dset, airval_record_tc_dset)`
Record processing entry for raw ai.
- `epicsExportAddress (dset, aaival_record_tc_dset)`
Record processing entry for aai.
- `epicsExportAddress (dset, bival_record_tc_dset)`
Record processing entry for bi.
- `epicsExportAddress (dset, birval_record_tc_dset)`
Record processing entry for raw bi.
- `epicsExportAddress (dset, longinval_record_tc_dset)`

- *Record processing entry for longin.*
 • **epicsExportAddress** (dset, mbbival_record_tc_dset)
Record processing entry for mbbi.
 - **epicsExportAddress** (dset, mbbirval_record_tc_dset)
Record processing entry for raw mbbi.
 - **epicsExportAddress** (dset, mbbiDirectval_record_tc_dset)
Record processing entry for mbbiDirect.
 - **epicsExportAddress** (dset, mbbiDirectrval_record_tc_dset)
Record processing entry for raw mbbiDirect.
 - **epicsExportAddress** (dset, stringinval_record_tc_dset)
Record processing entry for stringin.
 - **epicsExportAddress** (dset, waveformval_record_tc_dset)
Record processing entry for waveform.
 - **epicsExportAddress** (dset, aoval_record_tc_dset)
Record processing entry for ao.
 - **epicsExportAddress** (dset, aorval_record_tc_dset)
Record processing entry for raw ao.
 - **epicsExportAddress** (dset, aoval_record_tc_dset)
Record processing entry for aao.
 - **epicsExportAddress** (dset, boval_record_tc_dset)
Record processing entry for bo.
 - **epicsExportAddress** (dset, borval_record_tc_dset)
Record processing entry for raw bo.
 - **epicsExportAddress** (dset, longoutval_record_tc_dset)
Record processing entry for longout.
 - **epicsExportAddress** (dset, mbboval_record_tc_dset)
Record processing entry for mbbo.
 - **epicsExportAddress** (dset, mbborval_record_tc_dset)
Record processing entry for raw mbbo.
 - **epicsExportAddress** (dset, mbboDirectval_record_tc_dset)
Record processing entry for mbboDirect.
 - **epicsExportAddress** (dset, mbboDirectrval_record_tc_dset)
Record processing entry for raw mbboDirect.
 - **epicsExportAddress** (dset, stringoutval_record_tc_dset)
Record processing entry for stringout.

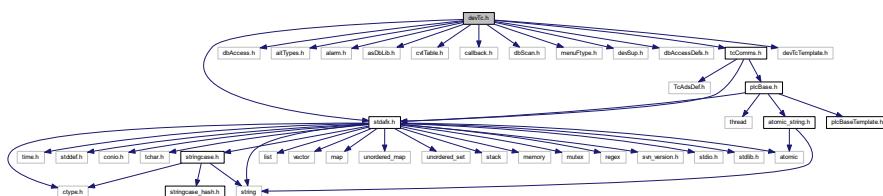
9.4.1 Detailed Description

Methods for TwinCAT/ADS device support.

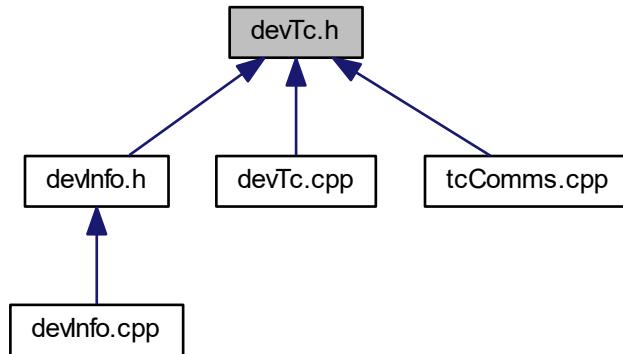
9.5 devTc.h File Reference

```
#include "stdafx.h"
#include "dbAccess.h"
#include "aitTypes.h"
#include "alarm.h"
#include "asDbLib.h"
#include "cvtTable.h"
```

```
#include "callback.h"
#include "dbScan.h"
#include "menuFType.h"
#include "devSup.h"
#include "dbAccessDefs.h"
#include "tcComms.h"
#include "devTcTemplate.h"
Include dependency graph for devTc.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [DevTc::register_devsup](#)
Device support registration.
- class [DevTc::EpicsInterface](#)
Epics interface class.
- struct [DevTc::epics_record_traits< RecType >](#)
Epics record traits.
- struct [DevTc::epics_record_traits< RecType >::traits_type](#)
Epics record type.
- struct [DevTc::devTcDeflo< RecType >](#)
Device support record.
- struct [DevTc::devTcDefIn< RecType >](#)
Device support input record.

- struct `DevTc::devTcDefOut< RecType >`
device support output record.
- struct `DevTc::devTcDefWaveformIn< RecType >`
device support waveform record.

Namespaces

- `DevTc`
Namespace for info device support.

Enumerations

- enum `DevTc::epics_record_enum` {
`DevTc::aaival` = 0, `DevTc::aaoval`, `DevTc::aival`, `DevTc::aoval`,
`DevTc::bival`, `DevTc::boval`, `DevTc::eventval`, `DevTc::histogramval`,
`DevTc::longinval`, `DevTc::longoutval`, `DevTc::mbbival`, `DevTc::mbboval`,
`DevTc::mbbiDirectval`, `DevTc::mbboDirectval`, `DevTc::stringinval`, `DevTc::stringoutval`,
`DevTc::waveformval`, `DevTc::airval`, `DevTc::aorval`, `DevTc::birval`,
`DevTc::borval`, `DevTc::mbbiDirectval`, `DevTc::mbboDirectval`, `DevTc::mbbirval`,
`DevTc::mbborval`, `DevTc::epics_record_enumEnd`, `DevTc::invalidval` = -1 }

Epics record type enum.

Functions

- void `DevTc::outRecordCallback` (`callbackPvt *pcallback`)
Callback for output record.
- bool `DevTc::linkRecord` (`std::stringcase name`, `dbCommon *pEpicsRecord`, `plc::BaseRecordPtr &pRecord`)
Link Record.
- bool `DevTc::linkTcRecord` (`dbCommon *pEpicsRecord`, `plc::BaseRecordPtr &pRecord`)
Link TwinCat Record.
- const `std::regex DevTc::tc_regex` ("((tc)::((\\b([0-9][1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.?)|(8[0-9][0-9]))|(\d{1,9})|(\d{1,9}):(\d{1,9})")
Regex for identifying TwinCAT records.

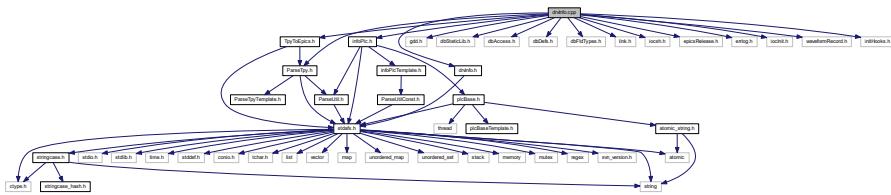
9.5.1 Detailed Description

Header which includes classes for TwinCAT/ADS device support.

9.6 drvInfo.cpp File Reference

```
#include "drvInfo.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
#include "infoPlc.h"
#include "gdd.h"
#include "dbStaticLib.h"
#include "dbAccess.h"
#include "dbDefs.h"
#include "dbFldTypes.h"
#include "link.h"
#include "iocsh.h"
#include "epicsRelease.h"
#include "errlog.h"
#include "iocInit.h"
#include "waveformRecord.h"
#include "initHooks.h"
```

Include dependency graph for drvInfo.cpp:



Classes

- class `DevInfo::epics_info_db_processing`
EPICS/Info db processing.

Namespaces

- DevInfo
Namespace for info device support.

TypeDefs

- `typedef std::pair< std::stringcase, std::stringcase > DevInfo::filename_rule_pair`
filename/rule pair

Functions

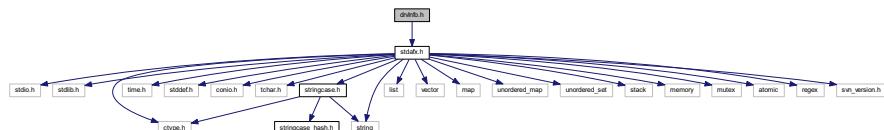
- void [DevInfo::infoLoadRecords](#) (const iocshArgBuf *args)
Info load records.
- void [DevInfo::infoSetScanRate](#) (const iocshArgBuf *args)
Info set scan rate.
- void [DevInfo::infoList](#) (const iocshArgBuf *args)
channel lists
- void [DevInfo::infoAlias](#) (const iocshArgBuf *args)
alias
- void [DevInfo::infoPrefix](#) (const iocshArgBuf *args)
tag prefix
- void [DevInfo::infoPrintVals](#) (const iocshArgBuf *args)
Info print vals.

9.6.1 Detailed Description

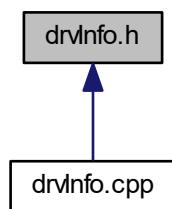
This contains functions for driver support for EPICS. These routines are used during the IOC initialization, and allow the IOC to initialize the TwinCAT interface during EPICS initialization.

9.7 drvInfo.h File Reference

```
#include "stdafx.h"
Include dependency graph for drvInfo.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [DevInfo::InfoRegisterTolocShell](#)

Register info commands.

Namespaces

- [DevInfo](#)

Namespace for info device support.

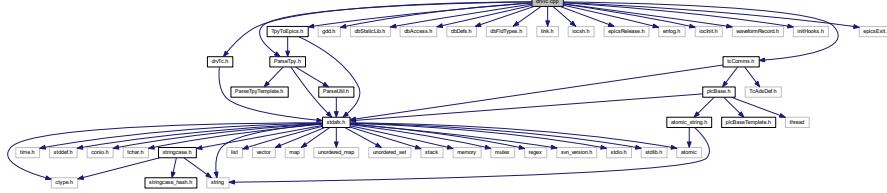
9.7.1 Detailed Description

Header which includes classes for accessing Info data from EPICS.

9.8 drvTc.cpp File Reference

```
#include "drvTc.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
#include "gdd.h"
#include "dbStaticLib.h"
#include "dbAccess.h"
#include "dbDefs.h"
#include "dbFldTypes.h"
#include "link.h"
#include "iocsh.h"
#include "epicsRelease.h"
#include "errlog.h"
#include "iocInit.h"
#include "waveformRecord.h"
#include "initHooks.h"
#include "tcComms.h"
#include "epicsExit.h"
```

Include dependency graph for drvTc.cpp:



Classes

- class [DevTc::epics_tc_db_processing](#)

EPICS/TCat db processing.

Namespaces

- **DevTc**

Namespace for info device support.

Typedefs

- **typedef std::tuple< std::stringcase, std::stringcase, epics_list_processing *, bool > DevTc::filename_rule_list_tuple**
Tuple for filnemae, rule and list processing.
- **typedef std::vector< filename_rule_list_tuple > DevTc::tc_listing_def**
List of tuples for filnemae, rule and list processing.
- **typedef std::tuple< std::stringcase, std::stringcase, epics_macrofiles_processing *, const char * > DevTc::dirname_arg_macro_tuple**
Tuple for directory name, argument and macro list processing.
- **typedef std::vector< dirname_arg_macro_tuple > DevTc::tc_macro_def**
List of tuples for directory name, argument and macro list processing.

Functions

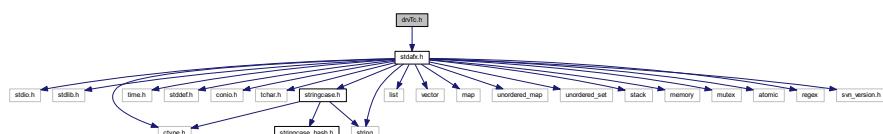
- **void DevTc::tcLoadRecords (const iocshArgBuf *args)**
TCat load records.
- **void DevTc::tcSetScanRate (const iocshArgBuf *args)**
TCat set scan rate.
- **void DevTc::tcList (const iocshArgBuf *args)**
channel lists
- **void DevTc::tcMacro (const iocshArgBuf *args)**
macro files
- **void DevTc::tcAlias (const iocshArgBuf *args)**
alias
- **void DevTc::tcPrintVals (const iocshArgBuf *args)**
TCat print vals.

9.8.1 Detailed Description

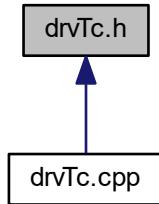
This contains functions for driver support for EPICS. These routines are used during the IOC initialization, and allow the IOC to initialize the TwinCAT interface during EPICS initialization.

9.9 drvTc.h File Reference

```
#include "stdafx.h"
Include dependency graph for drvTc.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [DevTc::tcRegisterToLocShell](#)
Register TC commands.

Namespaces

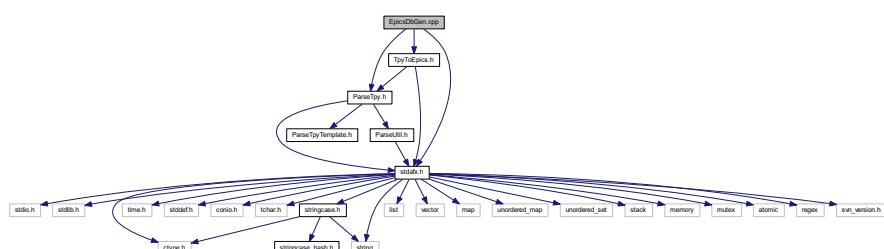
- [DevTc](#)
Namespace for info device support.

9.9.1 Detailed Description

Header which includes classes for accessing TwinCAT/ADS from EPICS.

9.10 EpicsDbGen.cpp File Reference

```
#include "stdaafx.h"
#include "ParseTpy.h"
#include "TpyToEpics.h"
Include dependency graph for EpicsDbGen.cpp:
```



Functions

- int [main](#) (int argc, char *argv[])

9.10.1 Detailed Description

Source for the main program that generates an EPICs .db file

9.10.2 Function Documentation

9.10.2.1 main()

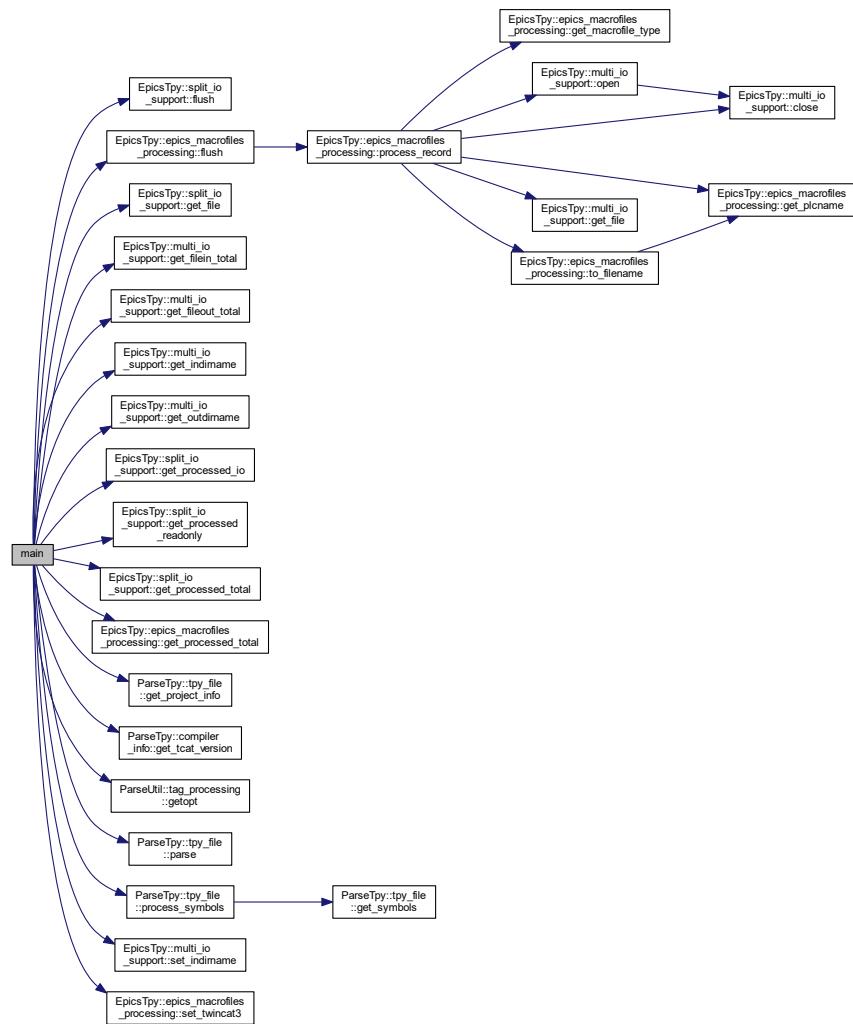
```
int main (
    int argc,
    char * argv[ ] )
```

Main program

Definition at line 18 of file EpicsDbGen.cpp.

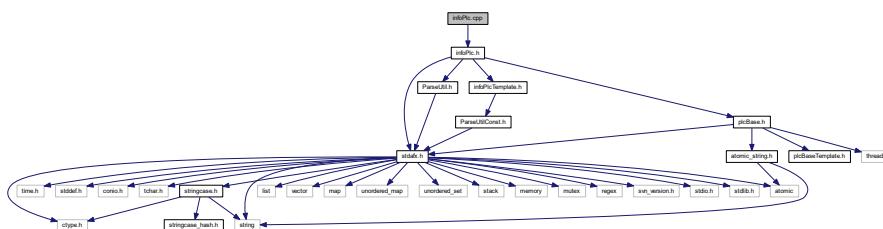
References `EpicsTpy::split_io_support::flush()`, `EpicsTpy::epics_macrofiles_processing::flush()`, `EpicsTpy::split_io_support::get_file()`, `EpicsTpy::multi_io_support::get_filein_total()`, `EpicsTpy::multi_io_support::get_fileout_total()`, `EpicsTpy::multi_io_support::get_indirname()`, `EpicsTpy::multi_io_support::get_outdirname()`, `EpicsTpy::split_io_support::get_processed_io()`, `EpicsTpy::split_io_support::get_processed_READONLY()`, `EpicsTpy::split_io_support::get_processed_total()`, `EpicsTpy::epics_macrofiles_processing::get_processed_total()`, `ParseTpy::tpy_file::get_project_info()`, `ParseTpy::compiler_info::get_tcat_version()`, `ParseUtil::tag_processing::getopt()`, `ParseTpy::tpy_file::parse()`, `ParseTpy::tpy_file::process_symbols()`, `EpicsTpy::multi_io_support::set_indirname()`, and `EpicsTpy::epics_macrofiles_processing::set_twincat3()`.

Here is the call graph for this function:



9.11 infoPlc.cpp File Reference

```
#include "infoPlc.h"
Include dependency graph for infoPlc.cpp:
```



Namespaces

- [InfoPlc](#)

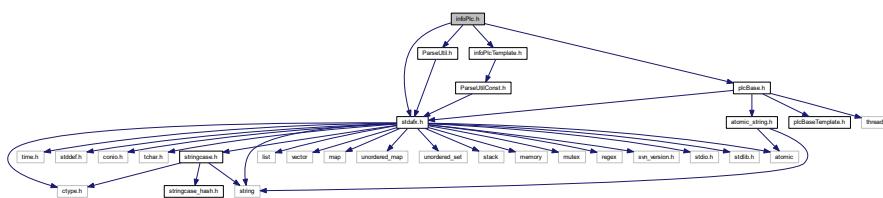
Namespace for Info communication.

9.11.1 Detailed Description

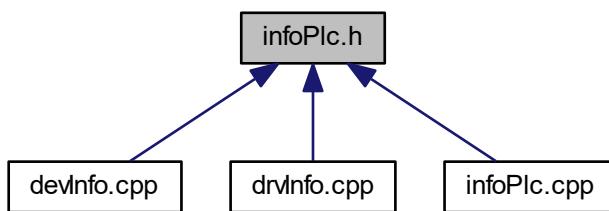
Defines methods for the info PLC.

9.12 infoPlc.h File Reference

```
#include "stdafx.h"
#include "ParseUtil.h"
#include "plcBase.h"
#include "infoPlcTemplate.h"
Include dependency graph for infoPlc.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct [InfoPlc::stat_value](#)
Statistic value.
- class [InfoPlc::BaseInfoItem](#)
Base info item.
- class [InfoPlc::SimpleInfoItem](#)

- Simple info tracker.*
- class [InfoPlc::HistogramInfoItem](#)

Histogram statistic.
 - class [InfoPlc::HistoryInfoItem](#)

History tracker.
 - class [InfoPlc::TimeInfoItem](#)

Timing tracker.
 - class [InfoPlc::InfoInterface](#)

Info interface.
 - class [InfoPlc::InfoPLC](#)

Info plc.

Namespaces

- [InfoPlc](#)

Namespace for Info communication.

Typedefs

- typedef std::vector< stat_value > [InfoPlc::stat_list](#)

Statistic list.
- typedef std::shared_ptr< BaseInfoItem > [InfoPlc::BaseInfoItemPtr](#)

Smart pointer to Base info items.
- typedef std::vector< BaseInfoItemPtr > [InfoPlc::BaseInfoList](#)

List of Base info items.

Variables

- const int [InfoPlc::default_scanrate](#) = 100

default PLC TwinCAT scan rate (100ms)
- const int [InfoPlc::minimum_scanrate](#) = 5

minimum PLC TwinCAT scan rate (5ms)
- const int [InfoPlc::maximum_scanrate](#) = 10000

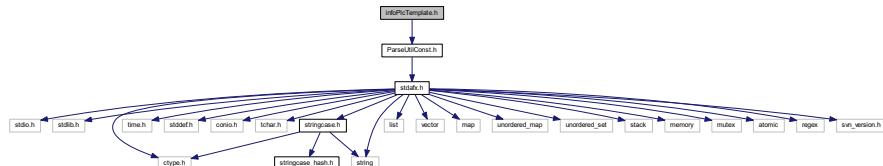
maximum PLC TwinCAT scan rate (10s)

9.12.1 Detailed Description

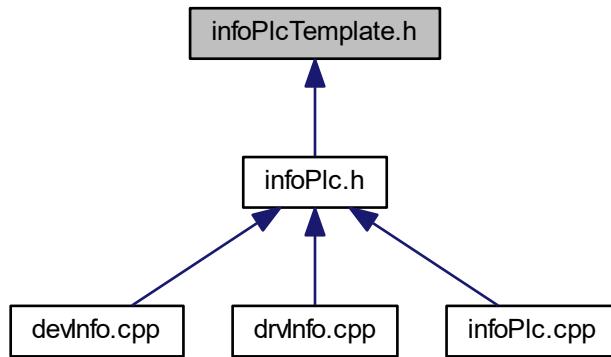
Header which includes classes for the info PLC.

9.13 infoPlcTemplate.h File Reference

```
#include "ParseUtilConst.h"
Include dependency graph for infoPlcTemplate.h:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- [InfoPlc](#)

Namespace for Info communication.

9.13.1 Detailed Description

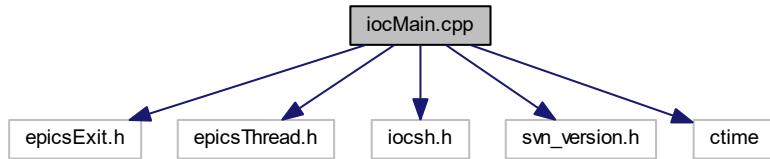
Header which includes classes for the info PLC.

9.14 iocMain.cpp File Reference

```
#include "epicsExit.h"
#include "epicsThread.h"
#include "iocsh.h"
#include "svn_version.h"
```

```
#include <ctime>
```

Include dependency graph for iocMain.cpp:



Functions

- `__declspec (dllexport) void stopTc(void)`
DLL import for stopTc.
- `int main (int argc, char *argv[])`
tcloc

9.14.1 Detailed Description

The main program for the TwinCAT IOC.

9.14.2 Function Documentation

9.14.2.1 main()

```
int main (
    int argc,
    char * argv[ ] )
```

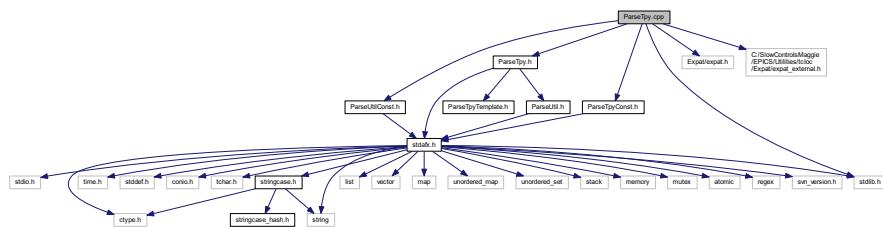
tcloc

Main program for tcloc

Definition at line 27 of file iocMain.cpp.

9.15 ParseTpy.cpp File Reference

```
#include "ParseTpy.h"
#include "ParseUtilConst.h"
#include "ParseTpyConst.h"
#include "Expat/expat.h"
Include dependency graph for ParseTpy.cpp:
```



Classes

- class [ParseTpy::parserinfo_type](#)

Namespaces

- [ParseTpy](#)

Namespace for parsing.

Macros

- #define [XML_STATIC](#)
Static linking.
- #define [XML_FMT_INT_MOD](#) "I"
Int format.

Functions

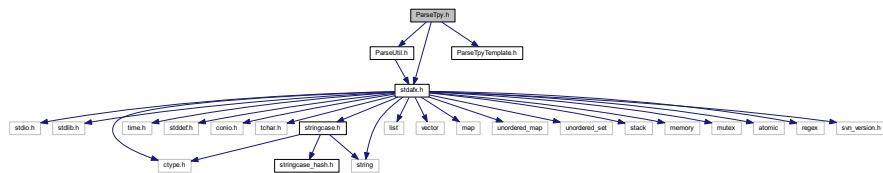
- bool [ParseTpy::compareNamesWoNamespace](#) (const std::stringcase &p1, const std::stringcase &p2)
- bool [ParseTpy::get_decoration](#) (const char **atts, unsigned int &decoration)
- bool [ParseTpy::get_pointer](#) (const char **atts)

9.15.1 Detailed Description

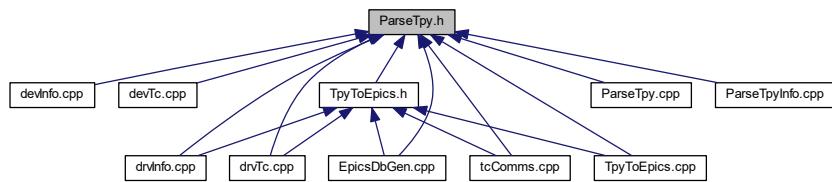
Methods to parse TwinCAT tpy file.

9.16 ParseTpy.h File Reference

```
#include "stdafx.h"
#include "ParseUtil.h"
#include "ParseTpyTemplate.h"
Include dependency graph for ParseTpy.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ParseTpy::ads_routing_info](#)
ADS routing information.
- class [ParseTpy::compiler_info](#)
Compiler information.
- class [ParseTpy::project_record](#)
Project information.
- class [ParseTpy::base_record](#)
Base record definition.
- class [ParseTpy::item_record](#)
item record
- class [ParseTpy::type_record](#)
- class [ParseTpy::type_map](#)
Type dictionary.
- class [ParseTpy::symbol_record](#)
Symbol record.
- class [ParseTpy::tpy_file](#)
Tpy file parsing.

Namespaces

- [ParseTpy](#)
Namespace for parsing.

Typedefs

- `typedef std::pair< int, int > ParseTpy::dimension`
- `typedef std::list< dimension > ParseTpy::dimensions`
- `typedef std::map< int, std::stringcase > ParseTpy::enum_map`
- `typedef std::pair< int, std::stringcase > ParseTpy::enum_pair`
- `typedef std::list< item_record > ParseTpy::item_list`
- `typedef std::multimap< unsigned int, type_record > ParseTpy::type_multipmap`
- `typedef std::list< symbol_record > ParseTpy::symbol_list`

Enumerations

- `enum ParseTpy::type_enum {
ParseTpy::unknown, ParseTpy::simple, ParseTpy::arraytype, ParseTpy::enumtype,
ParseTpy::structtype, ParseTpy::functionblock }`

Type enum.

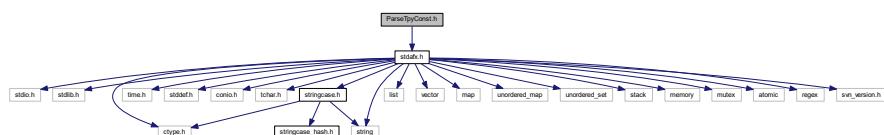
9.16.1 Detailed Description

Header which includes classes to parse a TwinCAT tpy file. It includes "ParseUtil.h" and "ParseTpyTemplate.h"

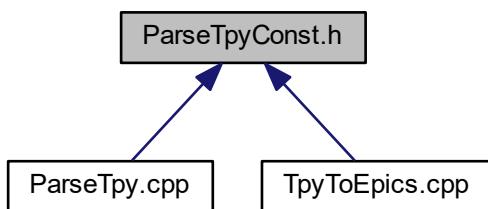
9.17 ParseTpyConst.h File Reference

```
#include "stdafx.h"
```

Include dependency graph for ParseTpyConst.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- [ParseUtil](#)
Namespace for parsing utilities.
- [ParseTpy](#)
Namespace for parsing.

Variables

- `const char *const ParseTpy::xmlPlcProjectInfo = "PlcProjectInfo"`
PLC project info.
- `const char *const ParseTpy::xmlProjectInfo = "ProjectInfo"`
Project info.
- `const char *const ParseTpy::xmlRoutingInfo = "RoutingInfo"`
Routing info.
- `const char *const ParseTpy::xmlCompilerInfo = "CompilerInfo"`
Compiler info.
- `const char *const ParseTpy::xmlAdsInfo = "AdsInfo"`
ADS info.
- `const char *const ParseTpy::xmlDataTypes = "DataTypes"`
Data types.
- `const char *const ParseTpy::xmlDataType = "DataType"`
Data type.
- `const char *const ParseTpy::xmlSymbols = "Symbols"`
Symbols.
- `const char *const ParseTpy::xmlSymbol = "Symbol"`
Symbol.
- `const char *const ParseTpy::xmlProperties = "Properties"`
Properties.
- `const char *const ParseTpy::xmlProperty = "Property"`
Property.
- `const char *const ParseTpy::xmlCompilerVersion = "CompilerVersion"`
Compiler version.
- `const char *const ParseTpy::xmlTwinCATVersion = "TwinCATVersion"`
TwinCAT version.
- `const char *const ParseTpy::xmlCpuFamily = "CpuFamily"`
CPU family.
- `const char *const ParseTpy::xmlNetId = "NetId"`
Net ID.
- `const char *const ParseTpy::xmlPort = "Port"`
Port.
- `const char *const ParseTpy::xmlTargetName = "TargetName"`
Target name.
- `const char *const ParseTpy::xmlName = "Name"`
Name.
- `const char *const ParseTpy::xmlType = "Type"`
Type.
- `const char *const ParseTpy::xmlAttrDecoration = "Decoration"`
Decoration.
- `const char *const ParseTpy::xmlAttrPointer = "Pointer"`

- const char *const ParseTpy::xmlIGroup = "IGroup"

I Group.
- const char *const ParseTpy::xmlIOffset = "IOffset"

I Offset.
- const char *const ParseTpy::xmlBitSize = "BitSize"

Bit size.
- const char *const ParseTpy::xmlBitOffs = "BitOffs"

Bit Offset.
- const char *const ParseTpy::xmlArrayInfo = "ArrayInfo"

Array info.
- const char *const ParseTpy::xmlArrayLBound = "LBound"

Lower bound.
- const char *const ParseTpy::xmlArrayElements = "Elements"

Elements.
- const char *const ParseTpy::xmlSubItem = "SubItem"

Sub item.
- const char *const ParseTpy::xmlFbInfo = "FbInfo"

Fb info.
- const char *const ParseTpy::xmlEnumInfo = "EnumInfo"

Enum info.
- const char *const ParseTpy::xmlEnumText = "Text"

Text.
- const char *const ParseTpy::xmlEnumEnum = "Enum"

Enum.
- const char *const ParseTpy::xmlEnumComment = "Comment"

Comment.
- const char *const ParseTpy::xmlValue = "Value"

Value.
- const char *const ParseTpy::xmlDesc = "Desc"

Description.
- const char *const ParseTpy::opcExport = "opc"

OPC.
- const char *const ParseTpy::opcProp = "opc_prop"

OPC property.
- const char *const ParseTpy::opcBracket = "["

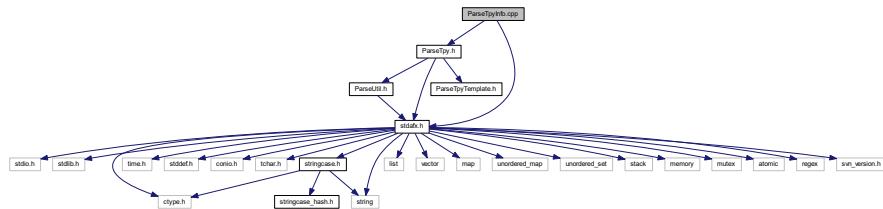
OPC bracket.

9.17.1 Detailed Description

Header which includes constants needed to parse a TwinCAT tpy file.

9.18 ParseTpyInfo.cpp File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
Include dependency graph for ParseTpyInfo.cpp:
```



Classes

- class [syminfo_processing](#)

Functions

- int [main](#) (int argc, char *argv[])
- tpyinfo*

9.18.1 Detailed Description

Source file for tpy parsing methods.

9.18.2 Function Documentation

9.18.2.1 main()

```
int main (
    int argc,
    char * argv[ ] )
```

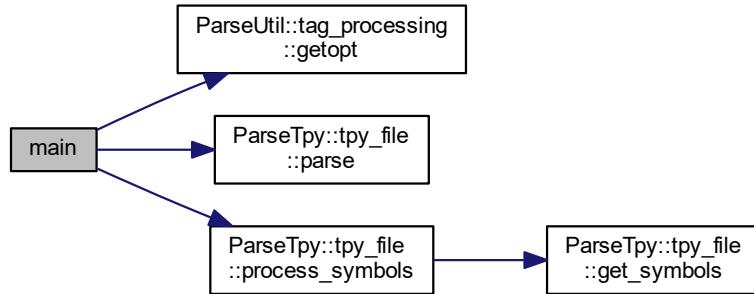
tpyinfo

Main program for tpyinfo

Definition at line 90 of file ParseTpyInfo.cpp.

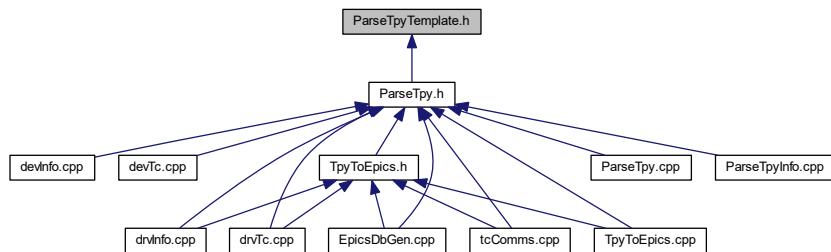
References `ParseUtil::tag_processing::getopt()`, `ParseTpy::tpy_file::parse()`, and `ParseTpy::tpy_file::process_symbols()`.

Here is the call graph for this function:



9.19 ParseTpypyTemplate.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- [ParseTpypy](#)

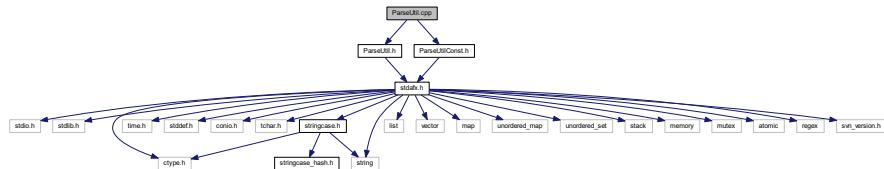
Namespace for parsing.

9.19.1 Detailed Description

Templates for functions for parsing tpy file.

9.20 ParseUtil.cpp File Reference

```
#include "ParseUtil.h"
#include "ParseUtilConst.h"
Include dependency graph for ParseUtil.cpp:
```



Namespaces

- **ParseUtil**

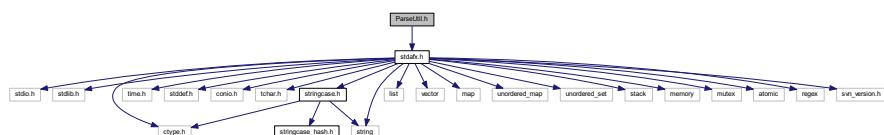
Namespace for parsing utilities.

9.20.1 Detailed Description

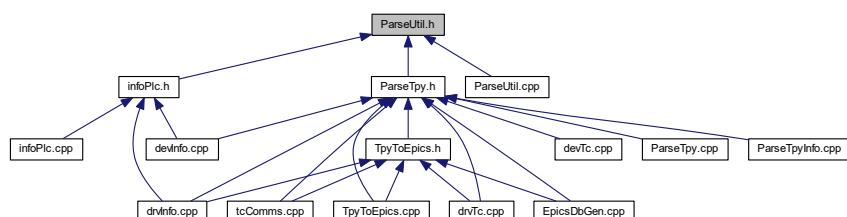
Utility methods to parsing.

9.21 ParseUtil.h File Reference

```
#include "stdafx.h"
Include dependency graph for ParseUtil.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [ParseUtil::optarg](#)
Optional arguments.
- class [ParseUtil::opc_list](#)
OPC list.
- class [ParseUtil::variable_name](#)
Variable name.
- class [ParseUtil::bit_location](#)
Bit location.
- class [ParseUtil::memory_location](#)
Memory location.
- class [ParseUtil::process_arg](#)
Arguments for processing.
- class [ParseUtil::tag_processing](#)
Tag processing selection.

Namespaces

- [ParseUtil](#)
Namespace for parsing utilities.

Typedefs

- [typedef std::map< int, std::stringcase > ParseUtil::property_map](#)
- [typedef std::pair< int, std::stringcase > ParseUtil::property_el](#)

Enumerations

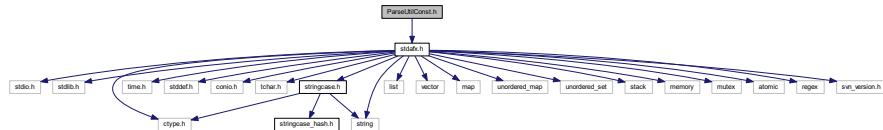
- enum [ParseUtil::opc_enum](#) { [ParseUtil::no_change](#), [ParseUtil::publish](#), [ParseUtil::silent](#) }
OPC state enum.
- enum [ParseUtil::process_type_enum](#) {
[ParseUtil::pt_invalid](#), [ParseUtil::pt_int](#), [ParseUtil::pt_real](#), [ParseUtil::pt_bool](#),
[ParseUtil::pt_string](#), [ParseUtil::pt_enum](#), [ParseUtil::pt_binary](#) }
Process type.
- enum [ParseUtil::process_tag_enum](#) { [ParseUtil::process_all](#), [ParseUtil::process_atomic](#), [ParseUtil::process_structured](#) }
Tag preoicessing enum.

9.21.1 Detailed Description

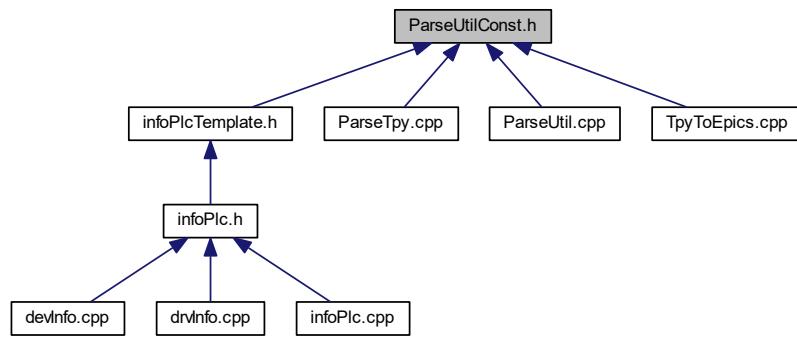
Header which includes utility classes for parsing. It includes "ParseUtilTemplate.h"

9.22 ParseUtilConst.h File Reference

```
#include "stdafx.h"
Include dependency graph for ParseUtilConst.h:
```



This graph shows which files directly or indirectly include this file:



Namespaces

- [ParseTpy](#)
Namespace for parsing.
- [ParseUtil](#)
Namespace for parsing utilities.

Variables

- const int [ParseUtil::OPC_PROP_CDT](#) = 1
- const int [ParseUtil::OPC_PROP_VALUE](#) = 2
- const int [ParseUtil::OPC_PROP_QUALITY](#) = 3
- const int [ParseUtil::OPC_PROP_TIME](#) = 4
- const int [ParseUtil::OPC_PROP_RIGHTS](#) = 5
- const int [ParseUtil::OPC_PROP_SCANRATE](#) = 6
- const int [ParseUtil::OPC_PROP_UNIT](#) = 100
- const int [ParseUtil::OPC_PROP_DESC](#) = 101
- const int [ParseUtil::OPC_PROP_HIEU](#) = 102
- const int [ParseUtil::OPC_PROP_LOEU](#) = 103
- const int [ParseUtil::OPC_PROP_HIRANGE](#) = 104
- const int [ParseUtil::OPC_PROP_LORANGE](#) = 105

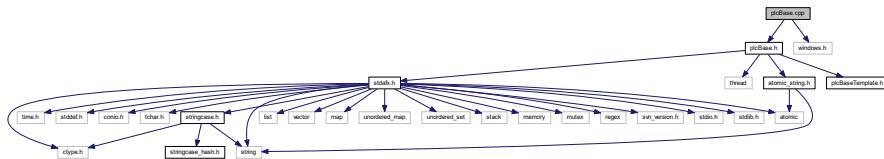
- const int ParseUtil::OPC_PROP_CLOSE = 106
- const int ParseUtil::OPC_PROP_OPEN = 107
- const int ParseUtil::OPC_PROP_TIMEZONE = 108
- const int ParseUtil::OPC_PROP_FGC = 201
- const int ParseUtil::OPC_PROP_BGC = 202
- const int ParseUtil::OPC_PROP_BLINK = 203
- const int ParseUtil::OPC_PROP_BMP = 204
- const int ParseUtil::OPC_PROP SND = 205
- const int ParseUtil::OPC_PROP_HTML = 206
- const int ParseUtil::OPC_PROP_AVI = 207
- const int ParseUtil::OPC_PROP_ALMSTAT = 300
- const int ParseUtil::OPC_PROP_ALMHELP = 301
- const int ParseUtil::OPC_PROP_ALMAREAS = 302
- const int ParseUtil::OPC_PROP_ALMPRIMARYAREA = 303
- const int ParseUtil::OPC_PROP_ALMCONDITION = 304
- const int ParseUtil::OPC_PROP_ALMLIMIT = 305
- const int ParseUtil::OPC_PROP_ALMDB = 306
- const int ParseUtil::OPC_PROP_ALMH = 307
- const int ParseUtil::OPC_PROP_ALMH = 308
- const int ParseUtil::OPC_PROP_ALML = 309
- const int ParseUtil::OPC_PROP_ALMLL = 310
- const int ParseUtil::OPC_PROP_ALMROC = 311
- const int ParseUtil::OPC_PROP_ALMDEV = 312
- const int ParseUtil::OPC_PROP_PREC = 8500
- const int ParseUtil::OPC_PROP_ZRST = 8510
- const int ParseUtil::OPC_PROP_FFST = 8525
- const int ParseUtil::OPC_PROP_RECTYPE = 8600
- const int ParseUtil::OPC_PROP_INOUT = 8601
- const char *const ParseUtil::OPC_PROP_INPUT = "input"
- const char *const ParseUtil::OPC_PROP_OUTPUT = "output"
- const int ParseUtil::OPC_PROP_TSE = 8602
- const int ParseUtil::OPC_PROP_PINI = 8603
- const int ParseUtil::OPC_PROP_DTYP = 8604
- const int ParseUtil::OPC_PROP_SERVER = 8610
- const int ParseUtil::OPC_PROP_PLNAME = 8611
- const int ParseUtil::OPC_PROP_ALIAS = 8620
- const int ParseUtil::OPC_PROP_ALMOSV = 8700
- const int ParseUtil::OPC_PROP_ALMZSV = 8701
- const int ParseUtil::OPC_PROP_ALMCOSV = 8702
- const int ParseUtil::OPC_PROP_ALMUNSV = 8703
- const int ParseUtil::OPC_PROP_ALMZRSV = 8710
- const int ParseUtil::OPC_PROP_ALMFSSV = 8725
- const int ParseUtil::OPC_PROP_ALMHHSV = 8727
- const int ParseUtil::OPC_PROP_ALMHHSV = 8728
- const int ParseUtil::OPC_PROP_ALMLSV = 8729
- const int ParseUtil::OPC_PROP_ALMLLSV = 8730

9.22.1 Detailed Description

Header which includes OPC constants.

9.23 plcBase.cpp File Reference

```
#include "plcBase.h"
#include <windows.h>
Include dependency graph for plcBase.cpp:
```



Classes

- struct [plc::scanner_thread_args](#)

Namespaces

- [plc](#)

Namespace for abstract plc functionality.

Functions

- template<>
bool [plc::reset_and_read](#) (DataValueTypeDef::atomic_bool &dirty, DataValueTypeDef::type_wstring &dest, DataValueTypeDef::atomic_string *source)
Reset and read.
- template<>
bool [plc::write_and_test](#) (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, DataValueTypeDef::atomic_wstring *dest, const DataValueTypeDef::type_string &source)
Write and test.
- VOID CALLBACK [plc::ScannerProc](#) (LPVOID lpArg, DWORD dwTimerLowValue, DWORD dwTimerHigh← Value)
- DWORD WINAPI [plc::scannerThread](#) (scanner_thread_args args)
- void [stopTc](#) (void)

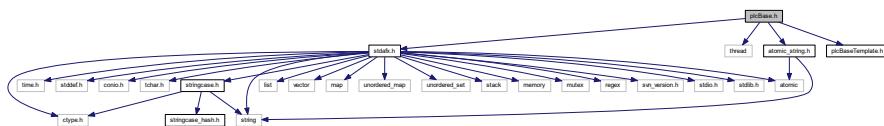
Stop TwinCAT.

9.23.1 Detailed Description

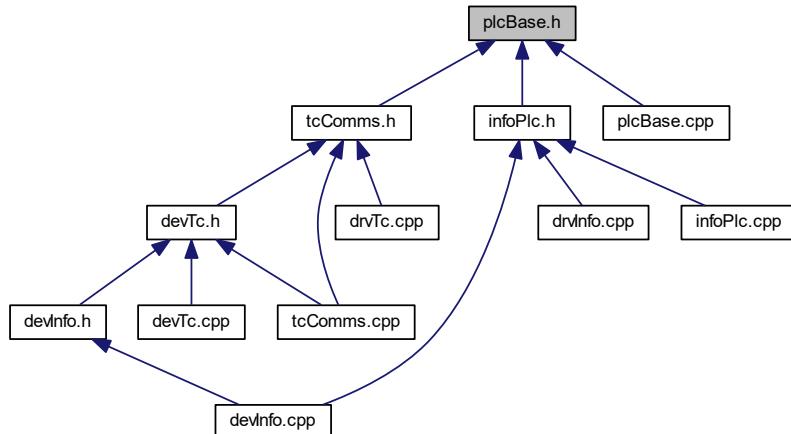
Defines methods for the internal record entry.

9.24 plcBase.h File Reference

```
#include "stdafx.h"
#include <thread>
#include "atomic_string.h"
#include "plcBaseTemplate.h"
Include dependency graph for plcBase.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [plc::Interface](#)
Abstract interface.
- struct [plc::DataValueTraits< T >](#)
Data value traits.
- struct [plc::DataValueTypeDef](#)
Collection of type definitions.
- class [plc::DataValue](#)
Data value.
- class [plc::BaseRecord](#)
Class for managing a tag/channel.
- class [plc::BasePLC](#)
Base PLC.
- class [plc::System](#)
System.

Namespaces

- `plc`

Namespace for abstract plc functionality.

Typedefs

- `typedef std::shared_ptr< BasePLC > plc::BasePLCPtr`
Smart pointer to PLC.
- `typedef std::unique_ptr< Interface > plc::InterfacePtr`
Smart pointer to interface.
- `typedef std::shared_ptr< BaseRecord > plc::BaseRecordPtr`
smart pointer to record
- `typedef std::unordered_map< std::stringcase, BaseRecordPtr > plc::BaseRecordList`
list of record
- `typedef std::map< std::stringcase, BasePLCPtr > plc::BasePLCList`
BasePLC map.

Enumerations

- `enum plc::data_type_enum {`
`plc::dtInvalid, plc::dtBool, plc::dtInt8, plc::dtUInt8,`
`plc::dtInt16, plc::dtUInt16, plc::dtInt32, plc::dtUInt32,`
`plc::dtInt64, plc::dtUInt64, plc::dtFloat, plc::dtDouble,`
`plc::dtString, plc::dtWString, plc::dtBinary }`
Data type enumeration.
- `enum plc::access_rights_enum { plc::read_only, plc::write_only, plc::read_write }`
Access rights enum.

Functions

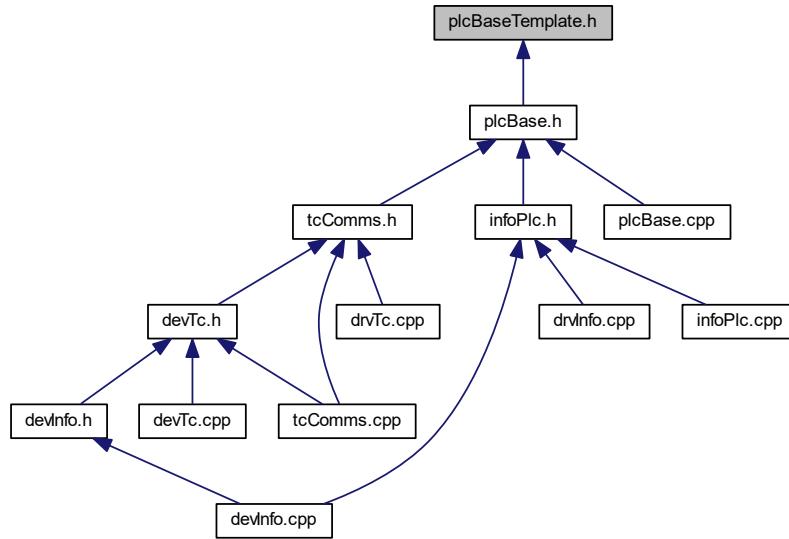
- `__declspec (dllexport) void __cdecl stopTc(void)`
Stop TwinCAT.

9.24.1 Detailed Description

Header which includes abstract base classes for defining an internal record entry for the IOC.

9.25 plcBaseTemplate.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- [plc](#)

Namespace for abstract plc functionality.

Functions

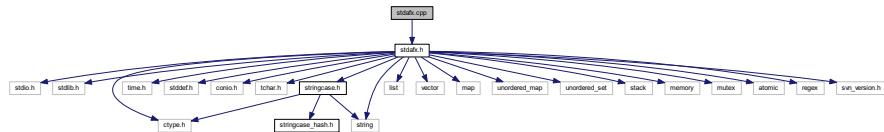
- template<typename T , typename U >
`bool plc::reset_and_read (DataValueTypeDef::atomic_bool &dirty, T &dest, U source)`
Reset and read.
- template<typename T >
`bool plc::reset_and_read (DataValueTypeDef::atomic_bool &dirty, T &dest, typename DataValueTraits< T >::traits_atomic *source)`
Reset and read.
- template<typename T , typename U >
`bool plc::write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, U dest, const T &source)`
Write and test.
- template<typename T >
`bool plc::write_and_test (DataValueTypeDef::atomic_bool &dirty, const DataValueTypeDef::atomic_bool &read_pending, DataValueTypeDef::atomic_bool &valid, typename DataValueTraits< T >::traits_atomic *dest, const T &source)`
Write and test.

9.25.1 Detailed Description

Header which includes templated methods for abstract record class and the DataValue class.

9.26 stdafx.cpp File Reference

```
#include "stdafx.h"
Include dependency graph for stdafx.cpp:
```

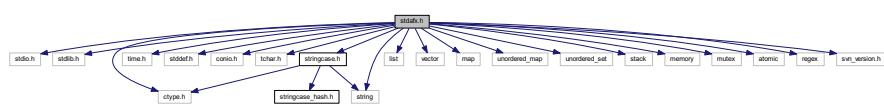


9.26.1 Detailed Description

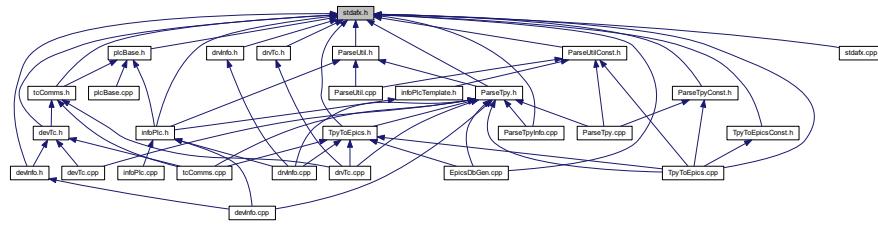
source file that includes just the standard includes EpicsDbGen.pch will be the pre-compiled header stdafx.obj will contain the pre-compiled type information

9.27 stdafx.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <time.h>
#include <stddef.h>
#include <conio.h>
#include <tchar.h>
#include <string>
#include "stringcase.h"
#include <list>
#include <vector>
#include <map>
#include <unordered_map>
#include <unordered_set>
#include <stack>
#include <memory>
#include <mutex>
#include <atomic>
#include <regex>
#include "svn_version.h"
Include dependency graph for stdafx.h:
```



This graph shows which files directly or indirectly include this file:

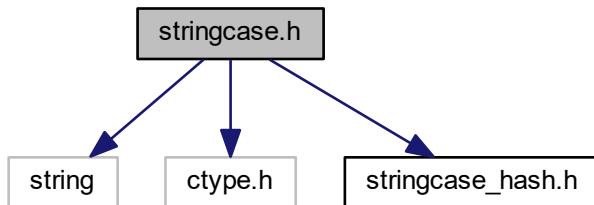


9.27.1 Detailed Description

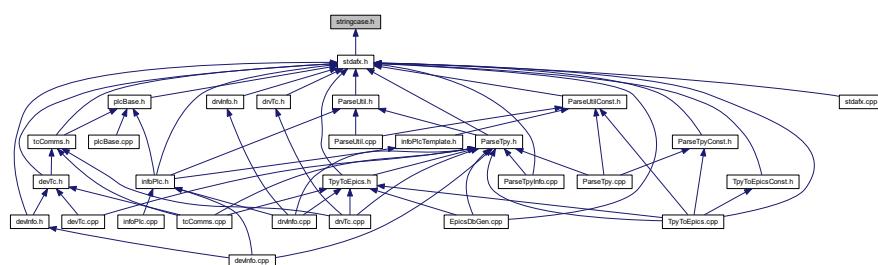
Include file for standard system include files, or project specific include files that are used frequently, but are changed infrequently

9.28 stringcase.h File Reference

```
#include <string>
#include <ctype.h>
#include "stringcase_hash.h"
Include dependency graph for stringcase.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct `std::case_char_traits`
case insensitive traits.
 - struct `std::case_wchar_traits`
case insensitive unicode traits.

TypeDefs

- `typedef std::basic_string< char, case_char_traits > std::stringcase`
case insensitive string.
 - `typedef std::basic_string< wchar_t, case_wchar_traits > std::wstringcase`
case insensitive string.

Functions

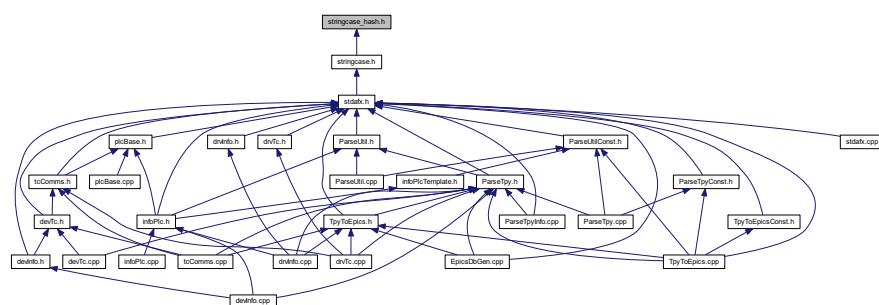
- int `std::strncasecmp` (const char *s1, const char *s2, size_t n)
case insensitive compare with maximum length
 - int `std::wcsncasewcmp` (const wchar_t *s1, const wchar_t *s2, size_t n)
case insensitive unicode compare with maximum length
 - void `std::trim_space` (`std::stringcase` &s)
 - void `std::trim_space` (`std::wstringcase` &s)
 - template<class Container , class String , class Predicate >
void `std::split_string` (Container &output, const String &input, const Predicate &pred, bool trimEmpty=true)
Splits a strings.

9.28.1 Detailed Description

Functions and classes for a case-insensitive string.

9.29 stringcase_hash.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

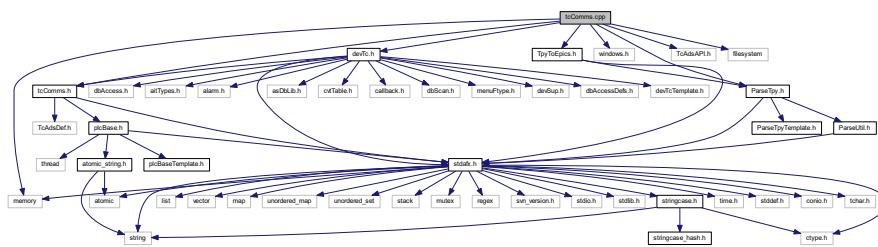
- struct `std::std::hash< std::stringcase >`
hash for case insensitive string.
- struct `std::std::hash< std::wstringcase >`
hash for case insensitive unicode string.

9.29.1 Detailed Description

Specialization for case sensitive strings.

9.30 tcComms.cpp File Reference

```
#include "tcComms.h"
#include "devTc.h"
#include "ParseTpy.h"
#include "windows.h"
#include "TpyToEpics.h"
#include "TcAdsAPI.h"
#include <memory>
#include <filesystem>
Include dependency graph for tcComms.cpp:
```



Namespaces

- `TcComms`
Namespace for TCat communication.

Functions

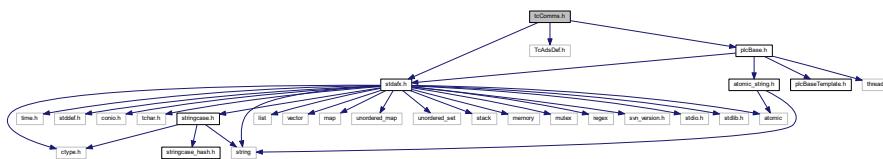
- void `TcComms::errorPrintf` (int nErr)
`errorPrintf`
- bool `TcComms::compByOffset` (BaseRecordPtr recA, BaseRecordPtr recB)
- void __stdcall `TcComms::ADSCallback` (AmsAddr *pAddr, AdsNotificationHeader *pNotification, unsigned long plcid)
- void __stdcall `TcComms::RouterCall` (long nReason)

9.30.1 Detailed Description

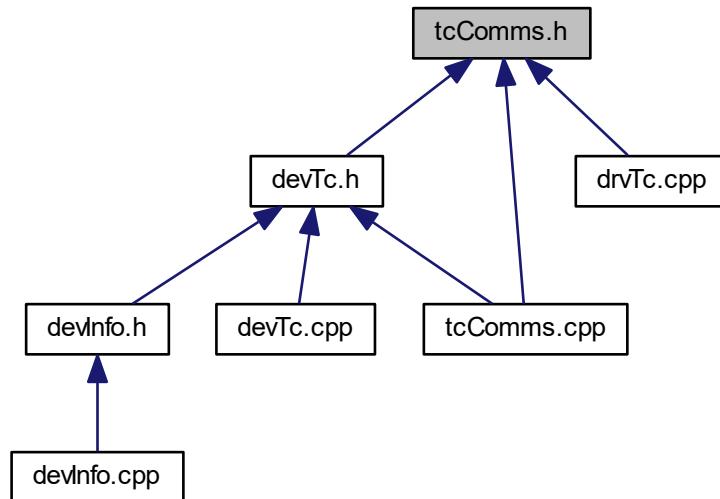
Defines methods for TwinCAT communication.

9.31 tcComms.h File Reference

```
#include "stdafx.h"
#include "TcAdsDef.h"
#include "plcBase.h"
Include dependency graph for tcComms.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct [TcComms::DataPar](#)
Memory location struct.
- class [TcComms::TCatInterface](#)
TCat interface class.
- class [TcComms::tcProcWrite](#)

- **TcComms**:
TwinCAT process write requests.
- class **TcComms::TcPLC**
TwinCAT PLC.
- class **TcComms::AmsRouterNotification**
AMS Router Notification.

Namespaces

- **TcComms**
Namespace for TCat communication.

Variables

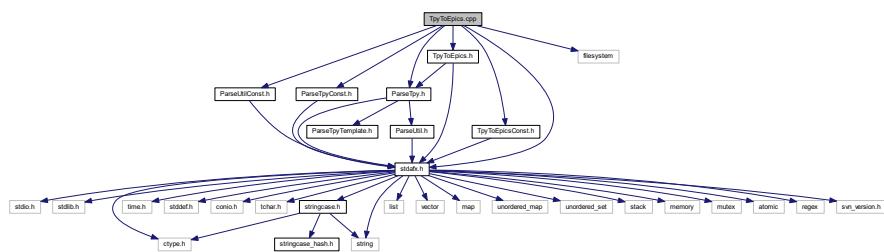
- const int **TcComms::MAX_REQ_SIZE** = 250000
maximum allowed request size (bytes)
- const int **TcComms::MAX_SINGLE_GAP_SIZE** = 50
maximum allowed size (bytes) of a memory gap within continuous request
- const double **TcComms::MAX_REL_GAP** = 0.25
(maximum allowed total gap size) / (current request size)
- const int **TcComms::MIN_REL_GAP_SIZE** = 100
minimum allowed relative gap size (bytes)
- const int **TcComms::default_scanrate** = 100
default PLC TwinCAT scan rate (100ms)
- const int **TcComms::minimum_scanrate** = 5
minimum PLC TwinCAT scan rate (5ms)
- const int **TcComms::maximum_scanrate** = 10000
maximum PLC TwinCAT scan rate (10s)
- const int **TcComms::default_multiple** = 10
default multiple for PLC EPICS scan rate (10)
- const int **TcComms::minimum_multiple** = 1
minimum multiple for PLC EPICS scan rate (1)
- const int **TcComms::maximum_multiple** = 200
maximum multiple for PLC EPICS scan rate (200)

9.31.1 Detailed Description

Header which includes classes to interface with the TCat system and manage TCat symbols.

9.32 TpyToEpics.cpp File Reference

```
#include "stdafx.h"
#include "ParseUtilConst.h"
#include "ParseTpyConst.h"
#include "ParseTpy.h"
#include "TpyToEpicsConst.h"
#include "TpyToEpics.h"
#include <filesystem>
Include dependency graph for TpyToEpics.cpp:
```



Namespaces

- EpicsTpy

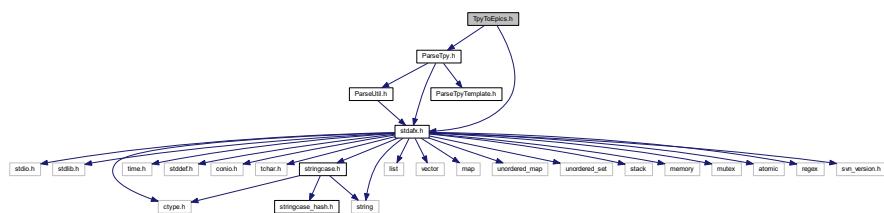
Namespace for tpy-db conversion.

9.32.1 Detailed Description

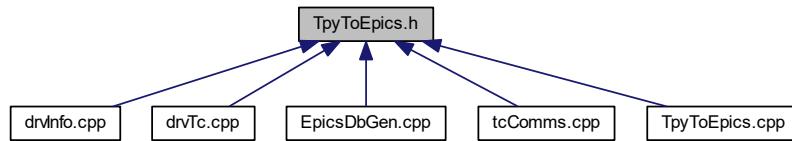
Source for methods that generate EPICs .db file from a .tpy file

9.33 TpyToEpics.h File Reference

```
#include "stdafx.h"
#include "ParseTpy.h"
Include dependency graph for TpyToEpics.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [EpicsTpy::replacement_rules](#)
Replacement rules.
- class [EpicsTpy::epics_conversion](#)
Epics conversion.
- class [EpicsTpy::split_io_support](#)
Split IO support.
- class [EpicsTpy::multi_io_support](#)
Multiple IO support.
- class [EpicsTpy::epics_list_processing](#)
List processing.
- struct [EpicsTpy::macro_info](#)
Macro information.
- struct [EpicsTpy::macro_record](#)
Macro record.
- class [EpicsTpy::epics_macrofiles_processing](#)
Macro file processing.
- class [EpicsTpy::epics_db_processing](#)
Epics database record processing

Namespaces

- [EpicsTpy](#)
Namespace for tpy-db conversion.

Typedefs

- `typedef std::map< std::stringcase, std::stringcase > EpicsTpy::replacement_table`
- `typedef std::vector< macro_info > EpicsTpy::macro_list`
- `typedef std::stack< macro_record > EpicsTpy::macro_stack`
- `typedef std::unordered_set< std::stringcase > EpicsTpy::filename_set`

Enumerations

- enum `EpicsTpy::tc_epics_conv` { `EpicsTpy::no_conversion`, `EpicsTpy::no_dot`, `EpicsTpy::ligo_std`, `EpicsTpy::ligo_vac` }

Conversion rules for TC/EPICS.
- enum `EpicsTpy::case_type` { `EpicsTpy::preserve_case`, `EpicsTpy::upper_case`, `EpicsTpy::lower_case` }

Case conversion rule enum.
- enum `EpicsTpy::io_filestat` { `EpicsTpy::io_filestat::closed`, `read`, `write` }

enum for file io
- enum `EpicsTpy::listing_type` { `EpicsTpy::listing_standard`, `EpicsTpy::listing_autoburt`, `EpicsTpy::listing_daqini` }

Listing type enum.
- enum `EpicsTpy::macrofile_type` { `EpicsTpy::macrofile_type::all`, `EpicsTpy::macrofile_type::fields`, `EpicsTpy::macrofile_type::erro` }
- enum `EpicsTpy::device_support_type` { `EpicsTpy::device_support_opc_name`, `EpicsTpy::device_support_tc_name` }

Device support enum.

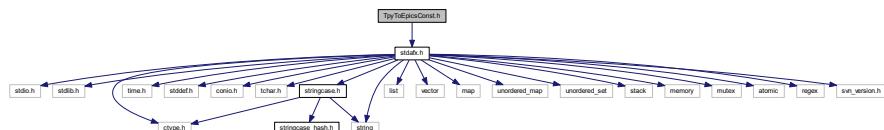
9.33.1 Detailed Description

Header which includes classes to convert a parsed TwinCAT tpy into an EPICS database. It includes "ParseTpy.h"

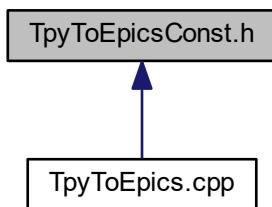
9.34 TpyToEpicsConst.h File Reference

```
#include "stdafx.h"
```

Include dependency graph for TpyToEpicsConst.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- [ParseTpy](#)
Namespace for parsing.
- [EpicsTpy](#)
Namespace for tpy-db conversion.

Variables

- const int [EpicsTpy::MAX_EPICS_CHANNEL](#) = 54
- const int [EpicsTpy::MAX_EPICS_DESC](#) = 40
- const int [EpicsTpy::MAX_EPICS_UNIT](#) = 15
- const char *const [EpicsTpy::EPICS_DB_EGU](#) = "EGU"
- const char *const [EpicsTpy::EPICS_DB_DESC](#) = "DESC"
- const char *const [EpicsTpy::EPICS_DB_HOPR](#) = "HOPR"
- const char *const [EpicsTpy::EPICS_DB_LOPR](#) = "LOPR"
- const char *const [EpicsTpy::EPICS_DB_DRVH](#) = "DRVH"
- const char *const [EpicsTpy::EPICS_DB_DRVL](#) = "DRVL"
- const char *const [EpicsTpy::EPICS_DB_ONAM](#) = "ONAM"
- const char *const [EpicsTpy::EPICS_DB_ZNAM](#) = "ZNAM"
- const char *const [EpicsTpy::EPICS_DB_PREC](#) = "PREC"
- const char *const [EpicsTpy::EPICS_DB_ZRST](#) [16]
- const char *const [EpicsTpy::EPICS_DB_ZRVL](#) [16]
- const char *const [EpicsTpy::EPICS_DB_SCAN](#) = "SCAN"
- const char *const [EpicsTpy::EPICS_DB_INP](#) = "INP"
- const char *const [EpicsTpy::EPICS_DB_OUT](#) = "OUT"
- const char *const [EpicsTpy::EPICS_DB_TSE](#) = "TSE"
- const char *const [EpicsTpy::EPICS_DB_PINI](#) = "PINI"
- const char *const [EpicsTpy::EPICS_DB_DTYP](#) = "DTYP"
- const char *const [EpicsTpy::EPICS_DB_OSV](#) = "OSV"
- const char *const [EpicsTpy::EPICS_DB_ZSV](#) = "ZSV"
- const char *const [EpicsTpy::EPICS_DB_COSV](#) = "COSV"
- const char *const [EpicsTpy::EPICS_DB_HIHI](#) = "HIHI"
- const char *const [EpicsTpy::EPICS_DB_HIGH](#) = "HIGH"
- const char *const [EpicsTpy::EPICS_DB_LOW](#) = "LOW"
- const char *const [EpicsTpy::EPICS_DB_LOLO](#) = "LOLO"
- const char *const [EpicsTpy::EPICS_DB_HYST](#) = "HYST"
- const char *const [EpicsTpy::EPICS_DB_HHSV](#) = "HHSV"
- const char *const [EpicsTpy::EPICS_DB_HSV](#) = "HSV"
- const char *const [EpicsTpy::EPICS_DB_LSV](#) = "LSV"
- const char *const [EpicsTpy::EPICS_DB_LLSV](#) = "LLSV"
- const char *const [EpicsTpy::EPICS_DB_NOALARM](#) = "NO_ALARM"
- const char *const [EpicsTpy::EPICS_DB_MINOR](#) = "MINOR"
- const char *const [EpicsTpy::EPICS_DB_MAJOR](#) = "MAJOR"
- const char *const [EpicsTpy::EPICS_DB_UNSV](#) = "UNSV"
- const char *const [EpicsTpy::EPICS_DB_ZRSV](#) [16]
- const char *const [EpicsTpy::EPICS_DB_FORBIDDEN](#) []
- const char *const [EpicsTpy::EPICS_DB_ALLOWED](#) []
- const char *const [EpicsTpy::EPICS_DB_NUMVAL](#) []
- const char *const [EpicsTpy::LIGODAQ_DATATYPE_NAME](#) = "datatype"
- const int [EpicsTpy::LIGODAQ_DATATYPE_FLOAT](#) = 4
- const int [EpicsTpy::LIGODAQ_DATATYPE_INT32](#) = 2
- const int [EpicsTpy::LIGODAQ_DATATYPE_DEFAULT](#) = LIGODAQ_DATATYPE_FLOAT
- const char *const [EpicsTpy::LIGODAQ_UNIT_NAME](#) = "units"
- const char *const [EpicsTpy::LIGODAQ_UNIT_NONE](#) = "none"
- const char *const [EpicsTpy::LIGODAQ_UNIT_DEFAULT](#) = LIGODAQ_UNIT_NONE
- const char *const [EpicsTpy::LIGODAQ_INI_HEADER](#)

9.34.1 Detailed Description

Header which includes EPICS constants needed to generate an EPICS db file from a parsed TwinCAT tpy file.

Index

- aaival
 - Device support for TwinCAT/ADS, [17](#)
- aooval
 - Device support for TwinCAT/ADS, [17](#)
- access_rights_enum
 - plc, [103](#)
- add
 - plc::BasePLC, [131](#)
 - plc::System, [302](#)
 - TcComms::tcProcWrite, [319](#)
- ADSCallback
 - TcComms, [110](#)
 - TcComms::TcPLC, [317](#)
- airval
 - Device support for TwinCAT/ADS, [17](#)
- aival
 - Device support for TwinCAT/ADS, [17](#)
- all
 - Classes for converting a parsed tpy, [71](#)
- aorval
 - Device support for TwinCAT/ADS, [17](#)
- aoaval
 - Device support for TwinCAT/ADS, [17](#)
- arraytype
 - Classes for describing a type, [33](#)
- atomic_string.h, [339](#)
- base_record
 - ParseTpy::base_record, [123, 124](#)
- BasePLCList
 - plc, [102](#)
- BasePLCPtr
 - plc, [102](#)
- BaseRecord
 - plc::BaseRecord, [140](#)
- BaseRecordList
 - plc, [103](#)
- BaseRecordPtr
 - plc, [103](#)
- birval
 - Device support for TwinCAT/ADS, [17](#)
- bival
 - Device support for TwinCAT/ADS, [17](#)
- borval
 - Device support for TwinCAT/ADS, [17](#)
- boval
 - Device support for TwinCAT/ADS, [17](#)
- case_type
 - Utility functions and classes, [67](#)
- Classes for converting a parsed tpy, [69](#)
 - all, [71](#)
 - device_support_opc_name, [70](#)
 - device_support_tc_name, [70](#)
 - device_support_type, [70](#)
 - errors, [71](#)
 - fields, [71](#)
 - filename_set, [69](#)
 - listing_autoburt, [71](#)
 - listing_daqini, [71](#)
 - listing_standard, [71](#)
 - listing_type, [70](#)
 - macro_list, [69](#)
 - macro_stack, [70](#)
 - macrofile_type, [71](#)
- Classes for describing a parser argument, [42](#)
 - process_all, [42](#)
 - process_atomic, [42](#)
 - process_structured, [42](#)
 - process_tag_enum, [42](#)
 - process_type_enum, [42](#)
 - pt_binary, [43](#)
 - pt_bool, [43](#)
 - pt_enum, [43](#)
 - pt_int, [43](#)
 - pt_invalid, [43](#)
 - pt_real, [43](#)
 - pt_string, [43](#)
- Classes for describing a structure element, [30](#)
 - dimension, [30](#)
 - dimensions, [30](#)
 - enum_map, [30](#)
 - enum_pair, [31](#)
 - item_list, [31](#)
- Classes for describing a symbol, [34](#)
 - symbol_list, [34](#)
- Classes for describing a type, [32](#)
 - arraytype, [33](#)
 - enumtype, [33](#)
 - functionblock, [33](#)
 - simple, [33](#)
 - structtype, [33](#)
 - type_enum, [32](#)
 - type_multipmap, [32](#)
 - unknown, [33](#)
- Classes for describing project information, [29](#)
- Classes for describing TC symbol, [65](#)
- Classes for describing the parser, [35](#)
- Classes for managing groups of TC symbols, [66](#)

closed
 Utility functions and classes, 68
 closePort
 TcComms::TcPLC, 314
 compare
 std::case_char_traits, 159
 std::case_wchar_traits, 162
 compareNamesWoNamespace
 ParseTpy, 98
 compByOffset
 TcComms, 111
 Constants related to read/write scanning, 28
 cyclesLeft
 TcComms::TcPLC, 317

 data_enum
 plc::DataValueTraits< T >, 199
 data_type_enum
 plc, 104
 DataValue
 plc::DataValue, 169
 Device support for TwinCAT/ADS, 13
 aaival, 17
 aaoval, 17
 airval, 17
 aival, 17
 aorval, 17
 aoval, 17
 birval, 17
 bival, 17
 borval, 17
 boval, 17
 epics_record_enum, 16
 epics_record_enumEnd, 17
 EpicsInterface, 17
 evental, 17
 get_callbackRequestPending, 17
 histogramval, 17
 invalidval, 17
 ioscanpvt, 19
 isCallback, 20
 isPassive, 20
 linkRecord, 18
 longinval, 17
 longoutval, 17
 mbbiDirectrval, 17
 mbbiDirectval, 17
 mbbirval, 17
 mbbival, 17
 mbboDirectrval, 17
 mbboDirectval, 17
 mbborval, 17
 mbboval, 17
 push, 19
 stringinval, 17
 stringoutval, 17
 the_register_devsup, 20
 value_ait_type, 20
 value_count, 21
 waveformval, 17

 device_support_opc_name
 Classes for converting a parsed tpy, 70
 device_support_tc_name
 Classes for converting a parsed tpy, 70
 device_support_type
 Classes for converting a parsed tpy, 70
 DevInfo, 85
 filename_rule_pair, 86
 linkInfoRecord, 86
 devInfo.cpp, 340
 devInfo.h, 342
 DevInfo::epics_info_db_processing, 225
 operator(), 226
 DevInfo::InfoRegisterTolocShell, 261
 DevInfo::register_info_devsup, 287
 DevTc, 87
 linkRecord, 89
 linkTcRecord, 90
 outRecordCallback, 91
 devTc.cpp, 343
 devTc.h, 344
 DevTc::devTcDefIn< RecType >, 201
 DevTc::devTcDeflo< RecType >, 203
 DevTc::devTcDefOut< RecType >, 205
 DevTc::devTcDefWaveformIn< RecType >, 206
 DevTc::epics_record_traits< RecType >, 242
 DevTc::epics_record_traits< RecType >::traits_type, 333
 DevTc::epics_tc_db_processing, 243
 operator(), 245
 process_list, 246
 process_lists, 247
 process_macro, 247
 process_macros, 248
 DevTc::EpicsInterface, 248
 DevTc::register_devsup, 285
 DevTc::tcRegisterTolocShell, 321
 dimension
 Classes for describing a structure element, 30
 dimensions
 Classes for describing a structure element, 30
 drvInfo.cpp, 347
 drvInfo.h, 348
 drvTc.cpp, 349
 drvTc.h, 350
 dtBinary
 plc, 104
 dtBool
 plc, 104
 dtDouble
 plc, 104
 dtFloat
 plc, 104
 dtInt16
 plc, 104
 dtInt32
 plc, 104

dtInt64
 plc, 104

dtInt8
 plc, 104

dtInvalid
 plc, 104

dtString
 plc, 104

dtUInt16
 plc, 104

dtUInt32
 plc, 104

dtUInt64
 plc, 104

dtUInt8
 plc, 104

dtWString
 plc, 104

enum_map
 Classes for describing a structure element, 30

enum_pair
 Classes for describing a structure element, 31

enumtype
 Classes for describing a type, 33

EPICS maximum length constants, 72
 MAX_EPICS_CHANNEL, 72
 MAX_EPICS_DESC, 72
 MAX_EPICS_UNIT, 72

EPICS record field names, 73
 EPICS_DB_COSV, 73
 EPICS_DB_DESC, 73
 EPICS_DB_DRVH, 74
 EPICS_DB_DRVL, 74
 EPICS_DB_DTYP, 74
 EPICS_DB_EGU, 74
 EPICS_DB_HHSV, 75
 EPICS_DB_HIGH, 75
 EPICS_DB_HIHI, 75
 EPICS_DB_HOPR, 75
 EPICS_DB_HSV, 76
 EPICS_DB_HYST, 76
 EPICS_DB_INP, 76
 EPICS_DB_LLSV, 76
 EPICS_DB_LOLO, 77
 EPICS_DB_LOPR, 77
 EPICS_DB_LOW, 77
 EPICS_DB_LSV, 77
 EPICS_DB_MAJOR, 78
 EPICS_DB_MINOR, 78
 EPICS_DB_NOALARM, 78
 EPICS_DB_ONAM, 78
 EPICS_DB_OSV, 79
 EPICS_DB_OUT, 79
 EPICS_DB_PINI, 79
 EPICS_DB_PREC, 79
 EPICS_DB_SCAN, 80
 EPICS_DB_TSE, 80
 EPICS_DB_UNSV, 80

 EPICS_DB_ZNAM, 80
 EPICS_DB_ZRST, 81
 EPICS_DB_ZRSV, 81
 EPICS_DB_ZRVL, 81
 EPICS_DB_ZSV, 81

 epics_conversion
 EpicsTpy::epics_conversion, 209, 210

 EPICS_DB_ALLOWED
 Lists of EPICS record field names, 82

 EPICS_DB_COSV
 EPICS record field names, 73

 EPICS_DB_DESC
 EPICS record field names, 73

 EPICS_DB_DRVH
 EPICS record field names, 74

 EPICS_DB_DRVL
 EPICS record field names, 74

 EPICS_DB_DTYP
 EPICS record field names, 74

 EPICS_DB_EGU
 EPICS record field names, 74

 EPICS_DB_FORBIDDEN
 Lists of EPICS record field names, 82

 EPICS_DB_HHSV
 EPICS record field names, 75

 EPICS_DB_HIGH
 EPICS record field names, 75

 EPICS_DB_HIHI
 EPICS record field names, 75

 EPICS_DB_HOPR
 EPICS record field names, 75

 EPICS_DB_HSV
 EPICS record field names, 76

 EPICS_DB_HYST
 EPICS record field names, 76

 EPICS_DB_INP
 EPICS record field names, 76

 EPICS_DB_LLSV
 EPICS record field names, 76

 EPICS_DB_LOLO
 EPICS record field names, 77

 EPICS_DB_LOPR
 EPICS record field names, 77

 EPICS_DB_LOW
 EPICS record field names, 77

 EPICS_DB_LSV
 EPICS record field names, 77

 EPICS_DB_MAJOR
 EPICS record field names, 78

 EPICS_DB_MINOR
 EPICS record field names, 78

 EPICS_DB_NOALARM
 EPICS record field names, 78

 EPICS_DB_NUMVAL
 Lists of EPICS record field names, 82

 EPICS_DB_ONAM
 EPICS record field names, 78

 EPICS_DB_OSV

EPICS record field names, 79
EPICS_DB_OUT
 EPICS record field names, 79
EPICS_DB_PINI
 EPICS record field names, 79
EPICS_DB_PREC
 EPICS record field names, 79
epics_db_processing
 EpicsTpy::epics_db_processing, 215
EPICS_DB_SCAN
 EPICS record field names, 80
EPICS_DB_TSE
 EPICS record field names, 80
EPICS_DB_UNSV
 EPICS record field names, 80
EPICS_DB_ZNAM
 EPICS record field names, 80
EPICS_DB_ZRST
 EPICS record field names, 81
EPICS_DB_ZRSV
 EPICS record field names, 81
EPICS_DB_ZRVL
 EPICS record field names, 81
EPICS_DB_ZSV
 EPICS record field names, 81
epics_list_processing
 EpicsTpy::epics_list_processing, 229
epics_macrofiles_processing
 EpicsTpy::epics_macrofiles_processing, 237
epics_record_enum
 Device support for TwinCAT/ADS, 16
epics_record_enumEnd
 Device support for TwinCAT/ADS, 17
EpicsDbGen.cpp, 351
 main, 352
EpicsInterface
 Device support for TwinCAT/ADS, 17
EpicsTpy, 91
EpicsTpy::epics_conversion, 207
 epics_conversion, 209, 210
 getopt, 210
 to_epics, 212
EpicsTpy::epics_db_processing, 213
 epics_db_processing, 215
 getopt, 216
 mygetopt, 217
 operator(), 218
 process_field_alarm, 220
 process_field_numeric, 221–223
 process_field_string, 223
EpicsTpy::epics_list_processing, 227
 epics_list_processing, 229
 getopt, 230
 mygetopt, 231
 operator(), 232
EpicsTpy::epics_macrofiles_processing, 234
 epics_macrofiles_processing, 237
 getopt, 238
 mygetopt, 239
 operator(), 240
 to_filename, 241
EpicsTpy::macro_info, 265
EpicsTpy::macro_record, 265
EpicsTpy::multi_io_support, 270
 getopt, 273
 multi_io_support, 272
EpicsTpy::replacement_rules, 288
EpicsTpy::split_io_support, 292
 getopt, 295
 increment, 296
 operator=, 297
 split_io_support, 294, 295
eq
 std::case_char_traits, 159
 std::case_wchar_traits, 162
erase
 plc::BasePLC, 132
errorPrintf
 TcComms, 111
errors
 Classes for converting a parsed tpy, 71
eventval
 Device support for TwinCAT/ADS, 17
fields
 Classes for converting a parsed tpy, 71
filename_rule_pair
 DevInfo, 86
filename_set
 Classes for converting a parsed tpy, 69
find
 plc::BasePLC, 132
for_each
 plc::BasePLC, 132
 plc::System, 302
functionblock
 Classes for describing a type, 33
Functions called by the EPICS base, 22
 infoAlias, 22
 infoList, 23
 infoLoadRecords, 23
 infoPrefix, 23
 infoPrintVals, 24
 infoSetScanRate, 24
 tcAlias, 25
 tcList, 25
 tcLoadRecords, 26
 tcMacro, 26
 tcPrintVals, 26
 tcSetScanRate, 27
get
 ParseTpy::ads_routing_info, 114, 115
 ParseUtil::memory_location, 268
 ParseUtil::process_arg, 283
get_callbackRequestPending
 Device support for TwinCAT/ADS, 17

get_decoration
 ParseTpy, 98

get_full
 ParseUtil::process_arg, 283

get_info
 InfoPlc::BaseInfoItem, 126

get_next
 plc::BasePLC, 133

get_pointer
 ParseTpy, 98

get_timestamp_unix
 plc::BasePLC, 134

getopt
 EpicsTpy::epics_conversion, 210
 EpicsTpy::epics_db_processing, 216
 EpicsTpy::epics_list_processing, 230
 EpicsTpy::epics_macrofiles_processing, 238
 EpicsTpy::multi_io_support, 273
 EpicsTpy::split_io_support, 295
 ParseUtil::tag_processing, 305

GetValid
 plc::DataValue, 169

histogramval
 Device support for TwinCAT/ADS, 17

increment
 EpicsTpy::split_io_support, 296

infoAlias
 Functions called by the EPICS base, 22

infoList
 Functions called by the EPICS base, 23

infoLoadRecords
 Functions called by the EPICS base, 23

InfoPlc, 94
 stat_list, 95

infoPlc.cpp, 353

infoPlc.h, 354

InfoPlc::BaseInfoItem, 125
 get_info, 126
 setup, 127

InfoPlc::HistogramInfoItem, 253

InfoPlc::HistoryInfoItem, 254
 setup, 256

InfoPlc::InfoInterface, 256

InfoPlc::InfoPLC, 258
 process_info, 259, 260

InfoPlc::SimpleInfoItem, 290
 setup, 291

InfoPlc::stat_value, 298

InfoPlc::TimeInfoItem, 321
 start, 323

infoPlcTemplate.h, 356

infoPrefix
 Functions called by the EPICS base, 23

infoPrintVals
 Functions called by the EPICS base, 24

infoSetScanRate
 Functions called by the EPICS base, 24

Init
 plc::DataValue, 170

Interface
 plc::Interface, 263

InterfacePtr
 plc, 103

invalidval
 Device support for TwinCAT/ADS, 17

io_filestat
 Utility functions and classes, 68

iocMain.cpp, 356
 main, 357

ioscanpvt
 Device support for TwinCAT/ADS, 19

isCallback
 Device support for TwinCAT/ADS, 20

isPassive
 Device support for TwinCAT/ADS, 20

item_list
 Classes for describing a structure element, 31

LIGO related constants, 83
 LIGODAQ_DATATYPE_DEFAULT, 83
 LIGODAQ_DATATYPE_FLOAT, 83
 LIGODAQ_DATATYPE_INT32, 83
 LIGODAQ_DATATYPE_NAME, 83
 LIGODAQ_INI_HEADER, 84
 LIGODAQ_UNIT_DEFAULT, 84
 LIGODAQ_UNIT_NAME, 84
 LIGODAQ_UNIT_NONE, 84

ligo_std
 Utility functions and classes, 68

ligo_vac
 Utility functions and classes, 68

LIGODAQ_DATATYPE_DEFAULT
 LIGO related constants, 83

LIGODAQ_DATATYPE_FLOAT
 LIGO related constants, 83

LIGODAQ_DATATYPE_INT32
 LIGO related constants, 83

LIGODAQ_DATATYPE_NAME
 LIGO related constants, 83

LIGODAQ_INI_HEADER
 LIGO related constants, 84

LIGODAQ_UNIT_DEFAULT
 LIGO related constants, 84

LIGODAQ_UNIT_NAME
 LIGO related constants, 84

LIGODAQ_UNIT_NONE
 LIGO related constants, 84

linkInfoRecord
 DevInfo, 86

linkRecord
 Device support for TwinCAT/ADS, 18
 DevTc, 89

linkTcRecord
 DevTc, 90

listing_autoburt
 Classes for converting a parsed tpy, 71

listing_daqini
 Classes for converting a parsed tpy, 71

listing_standard
 Classes for converting a parsed tpy, 71

listing_type
 Classes for converting a parsed tpy, 70

Lists of EPICS record field names, 82
 EPICS_DB_ALLOWED, 82
 EPICS_DB_FORBIDDEN, 82
 EPICS_DB_NUMVAL, 82

longinval
 Device support for TwinCAT/ADS, 17

longoutval
 Device support for TwinCAT/ADS, 17

lower_case
 Utility functions and classes, 68

lt
 std::case_char_traits, 160
 std::case_wchar_traits, 163

macro_list
 Classes for converting a parsed tpy, 69

macro_stack
 Classes for converting a parsed tpy, 70

macrofile_type
 Classes for converting a parsed tpy, 71

main
 EpicsDbGen.cpp, 352
 iocMain.cpp, 357
 ParseTpyInfo.cpp, 363

MAX_EPICS_CHANNEL
 EPICS maximum length constants, 72

MAX_EPICS_DESC
 EPICS maximum length constants, 72

MAX_EPICS_UNIT
 EPICS maximum length constants, 72

mbbiDirectrval
 Device support for TwinCAT/ADS, 17

mbbiDirectval
 Device support for TwinCAT/ADS, 17

mbbirval
 Device support for TwinCAT/ADS, 17

mbbival
 Device support for TwinCAT/ADS, 17

mbboDirectrval
 Device support for TwinCAT/ADS, 17

mbboDirectval
 Device support for TwinCAT/ADS, 17

mbborval
 Device support for TwinCAT/ADS, 17

mbboval
 Device support for TwinCAT/ADS, 17

memory_location
 ParseUtil::memory_location, 267, 268

multi_io_support
 EpicsTpy::multi_io_support, 272

mygetopt
 EpicsTpy::epics_db_processing, 217
 EpicsTpy::epics_list_processing, 231

 EpicsTpy::epics_macrofiles_processing, 239

mysize
 plc::DataValue, 197

name_parse
 ParseTpy::parserinfo_type, 280

ne
 std::case_char_traits, 160
 std::case_wchar_traits, 163

no_change
 OPC related functions and classes, 41

no_conversion
 Utility functions and classes, 68

no_dot
 Utility functions and classes, 68

OPC property constants, 44
 OPC_PROP_ALIAS, 45
 OPC_PROP_ALMAREAS, 45
 OPC_PROP_ALMCONDITION, 45
 OPC_PROP_ALMCOSV, 45
 OPC_PROP_ALMDB, 46
 OPC_PROP_ALMDEV, 46
 OPC_PROP_ALMFFSV, 46
 OPC_PROP_ALMH, 46
 OPC_PROP_ALMHELP, 47
 OPC_PROP_ALMH, 47
 OPC_PROP_ALMHHSV, 47
 OPC_PROP_ALMHSV, 47
 OPC_PROP_ALML, 48
 OPC_PROP_ALMLIMIT, 48
 OPC_PROP_ALMLL, 48
 OPC_PROP_ALMLLSV, 48
 OPC_PROP_ALMLSV, 49
 OPC_PROP_ALMOSV, 49
 OPC_PROP_ALMPRIMARYAREA, 49
 OPC_PROP_ALMROC, 49
 OPC_PROP_ALMSTAT, 49
 OPC_PROP_ALMUNSV, 50
 OPC_PROP_ALMZRSV, 50
 OPC_PROP_ALMZSV, 50
 OPC_PROP_AVI, 50
 OPC_PROP_BGC, 50
 OPC_PROP_BLINK, 51
 OPC_PROP_BMP, 51
 OPC_PROP_CDT, 51
 OPC_PROP_CLOSE, 51
 OPC_PROP_DESC, 51
 OPC_PROP_DTYP, 52
 OPC_PROP_FFST, 52
 OPC_PROP_FGC, 52
 OPC_PROP_HIEU, 52
 OPC_PROP_HIRANGE, 53
 OPC_PROP_HTML, 53
 OPC_PROP_INOUT, 53
 OPC_PROP_INPUT, 53
 OPC_PROP_LOEU, 54
 OPC_PROP_LORANGE, 54
 OPC_PROP_OPEN, 54

OPC_PROP_OUTPUT, 54
OPC_PROP_PINI, 55
OPC_PROP_PLNAME, 55
OPC_PROP_PREC, 55
OPC_PROP_QUALITY, 55
OPC_PROP_RECTYPE, 56
OPC_PROP_RIGHTS, 56
OPC_PROP_SCANRATE, 56
OPC_PROP_SERVER, 56
OPC_PROP SND, 56
OPC_PROP_TIME, 57
OPC_PROP_TIMEZONE, 57
OPC_PROP_TSE, 57
OPC_PROP_UNIT, 57
OPC_PROP_VALUE, 57
OPC_PROP_ZRST, 58
OPC related functions and classes, 40
 no_change, 41
 opc_enum, 40
 property_el, 40
 property_map, 40
 publish, 41
 silent, 41
OPC tpy file constants, 38
opc_enum
 OPC related functions and classes, 40
OPC_PROP_ALIAS
 OPC property constants, 45
OPC_PROP_ALMAREAS
 OPC property constants, 45
OPC_PROP_ALMCCONDITION
 OPC property constants, 45
OPC_PROP_ALMCOSV
 OPC property constants, 45
OPC_PROP_ALMDB
 OPC property constants, 46
OPC_PROP_ALMDEV
 OPC property constants, 46
OPC_PROP_ALMFFSV
 OPC property constants, 46
OPC_PROP_ALMH
 OPC property constants, 46
OPC_PROP_ALMHELP
 OPC property constants, 47
OPC_PROP_ALMH
 OPC property constants, 47
OPC_PROP_ALMHHSV
 OPC property constants, 47
OPC_PROP_ALMHHSV
 OPC property constants, 47
OPC_PROP_ALMLIMIT
 OPC property constants, 48
OPC_PROP_ALMLL
 OPC property constants, 48
OPC_PROP_ALMLLSV
 OPC property constants, 48
OPC_PROP_ALMLSV
 OPC property constants, 49
OPC_PROP_ALMOSV
 OPC property constants, 49
OPC_PROP_ALMPPRIMARYAREA
 OPC property constants, 49
OPC_PROP_ALMROC
 OPC property constants, 49
OPC_PROP_ALMSTAT
 OPC property constants, 49
OPC_PROP_ALMUNSV
 OPC property constants, 50
OPC_PROP_ALMZRSV
 OPC property constants, 50
OPC_PROP_ALMZSV
 OPC property constants, 50
OPC_PROP_AVI
 OPC property constants, 50
OPC_PROP_BGC
 OPC property constants, 50
OPC_PROP_BLINK
 OPC property constants, 51
OPC_PROP_BMP
 OPC property constants, 51
OPC_PROP_CDT
 OPC property constants, 51
OPC_PROP_CLOSE
 OPC property constants, 51
OPC_PROP_DESC
 OPC property constants, 51
OPC_PROP_DTYP
 OPC property constants, 52
OPC_PROP_FFST
 OPC property constants, 52
OPC_PROP_FGC
 OPC property constants, 52
OPC_PROP_HIEU
 OPC property constants, 52
OPC_PROP_HIRANGE
 OPC property constants, 53
OPC_PROP_HTML
 OPC property constants, 53
OPC_PROP_INOUT
 OPC property constants, 53
OPC_PROP_INPUT
 OPC property constants, 53
OPC_PROP_LOEU
 OPC property constants, 54
OPC_PROP_LORANGE
 OPC property constants, 54
OPC_PROP_OPEN
 OPC property constants, 54
OPC_PROP_OUTPUT
 OPC property constants, 54
OPC_PROP_PINI
 OPC property constants, 55
OPC_PROP_PLNAME
 OPC property constants, 55

OPC_PROP_PREC
 OPC property constants, 55

OPC_PROP_QUALITY
 OPC property constants, 55

OPC_PROP_RECTYPE
 OPC property constants, 56

OPC_PROP_RIGHTS
 OPC property constants, 56

OPC_PROP_SCANRATE
 OPC property constants, 56

OPC_PROP_SERVER
 OPC property constants, 56

OPC_PROP SND
 OPC property constants, 56

OPC_PROP_TIME
 OPC property constants, 57

OPC_PROP_TIMEZONE
 OPC property constants, 57

OPC_PROP_TSE
 OPC property constants, 57

OPC_PROP_UNIT
 OPC property constants, 57

OPC_PROP_VALUE
 OPC property constants, 57

OPC_PROP_ZRST
 OPC property constants, 58

operator()
 DevInfo::epics_info_db_processing, 226
 DevTc::epics_tc_db_processing, 245
 EpicsTpy::epics_db_processing, 218
 EpicsTpy::epics_list_processing, 232
 EpicsTpy::epics_macrofiles_processing, 240
 std::std::hash< std::stringcase >, 251
 std::std::hash< std::wstringcase >, 252

operator=
 EpicsTpy::split_io_support, 297
 std::atomic_string< stringT >, 121

optarg
 ParseUtil::optarg, 276

optimizeRequests
 TcComms::TcPLC, 315

Option processing, 39

outRecordCallback
 DevTc, 91

parse
 ParseUtil::optarg, 277

parse_finish
 ParseTpy::tpy_file, 325

ParseTpy, 95
 compareNamesWoNamespace, 98
 get_decoration, 98
 get_pointer, 98

ParseTpy.cpp, 358

ParseTpy.h, 359

ParseTpy::ads_routing_info, 113
 get, 114, 115
 set, 115

ParseTpy::base_record, 122
 base_record, 123, 124

ParseTpy::compiler_info, 163

ParseTpy::item_record, 263

ParseTpy::parserinfo_type, 278
 name_parse, 280

ParseTpy::project_record, 284

ParseTpy::symbol_record, 299

ParseTpy::tpy_file, 323
 parse_finish, 325
 process_array, 325
 process_symbols, 327
 process_type_tree, 327, 329, 331

ParseTpy::type_map, 333

ParseTpy::type_record, 334

ParseTpyConst.h, 360

ParseTpyInfo.cpp, 363
 main, 363

ParseTpyTemplate.h, 364

ParseUtil, 99

ParseUtil.cpp, 365

ParseUtil.h, 365

ParseUtil::bit_location, 157

ParseUtil::memory_location, 266
 get, 268
 memory_location, 267, 268
 set, 269
 set_section, 269

ParseUtil::opc_list, 274

ParseUtil::optarg, 275
 optarg, 276
 parse, 277

ParseUtil::process_arg, 281
 get, 283
 get_full, 283
 process_arg, 282

ParseUtil::tag_processing, 303
 getopt, 305
 tag_processing, 304, 305

ParseUtil::variable_name, 336

ParseUtilConst.h, 367

plc, 101
 access_rights_enum, 103
 BasePLCList, 102
 BasePLCPtr, 102
 BaseRecordList, 103
 BaseRecordPtr, 103
 data_type_enum, 104
 dtBinary, 104
 dtBool, 104
 dtDouble, 104
 dtFloat, 104
 dtInt16, 104
 dtInt32, 104
 dtInt64, 104
 dtInt8, 104
 dtInvalid, 104
 dtString, 104
 dtUInt16, 104

dtUInt32, 104
dtUInt64, 104
dtUInt8, 104
dtWString, 104
InterfacePtr, 103
read_only, 104
read_write, 104
reset_and_read, 104, 105
ScannerProc, 105
scannerThread, 106
write_and_test, 107, 108
write_only, 104
plc::BasePLC, 128
 add, 131
 erase, 132
 find, 132
 for_each, 132
 get_next, 133
 get_timestamp_unix, 134
 plc_data_set_valid, 134
 records, 136
 reserve, 135
 start, 135
 user_data_set_valid, 135
plc::BaseRecord, 137
 BaseRecord, 140
 PlcGetValid, 140
 PlcPush, 141
 PlcRead, 142, 144
 PlcReadBinary, 145
 PlcSetValid, 145
 PlcWrite, 146, 147
 PlcWriteBinary, 148
 UserGetValid, 149
 UserPush, 149
 UserRead, 150–152
 UserReadBinary, 152
 UserSetValid, 153
 UserWrite, 154, 155
 UserWriteBinary, 156
plc::DataValue, 166
 DataValue, 169
 GetValid, 169
 Init, 170
 mysize, 197
 PlcGetValid, 171
 PlcRead, 171–173
 PlcReadBinary, 174
 PlcSetValid, 175
 PlcWrite, 175–177
 PlcWriteBinary, 178
 Read, 179–181
 ReadBinary, 182
 SetValid, 183
 UserGetValid, 184
 UserRead, 184–186
 UserReadBinary, 187
 UserSetValid, 188
 UserWrite, 188–190
 UserWriteBinary, 191
 Write, 192–195
 WriteBinary, 195
plc::DataValueTraits< T >, 198
 data_enum, 199
plc::DataValueTypeDef, 199
plc::Interface, 262
 Interface, 263
plc::scanner_thread_args, 289
plc::System, 300
 add, 302
 for_each, 302
plc_data_set_valid
 plc::BasePLC, 134
plcBase.cpp, 369
plcBase.h, 370
plcBaseTemplate.h, 372
PlcGetValid
 plc::BaseRecord, 140
 plc::DataValue, 171
PlcPush
 plc::BaseRecord, 141
PlcRead
 plc::BaseRecord, 142, 144
 plc::DataValue, 171–173
PlcReadBinary
 plc::BaseRecord, 145
 plc::DataValue, 174
PlcSetValid
 plc::BaseRecord, 145
 plc::DataValue, 175
PlcWrite
 plc::BaseRecord, 146, 147
 plc::DataValue, 175–177
PlcWriteBinary
 plc::BaseRecord, 148
 plc::DataValue, 178
preserve_case
 Utility functions and classes, 68
printTCatVal
 TcComms::TCatInterface, 310
process_all
 Classes for describing a parser argument, 42
process_arg
 ParseUtil::process_arg, 282
process_array
 ParseTpy::tpy_file, 325
process_atomic
 Classes for describing a parser argument, 42
process_field_alarm
 EpicsTpy::epics_db_processing, 220
process_field_numeric
 EpicsTpy::epics_db_processing, 221–223
process_field_string
 EpicsTpy::epics_db_processing, 223
process_info
 InfoPlc::InfoPLC, 259, 260

process_list
 DevTc::epics_tc_db_processing, 246

process_lists
 DevTc::epics_tc_db_processing, 247

process_macro
 DevTc::epics_tc_db_processing, 247

process_macros
 DevTc::epics_tc_db_processing, 248

process_structured
 Classes for describing a parser argument, 42

process_symbols
 ParseTpy::tpy_file, 327

process_tag_enum
 Classes for describing a parser argument, 42

process_type_enum
 Classes for describing a parser argument, 42

process_type_tree
 ParseTpy::tpy_file, 327, 329, 331

property_el
 OPC related functions and classes, 40

property_map
 OPC related functions and classes, 40

pt_binary
 Classes for describing a parser argument, 43

pt_bool
 Classes for describing a parser argument, 43

pt_enum
 Classes for describing a parser argument, 43

pt_int
 Classes for describing a parser argument, 43

pt_invalid
 Classes for describing a parser argument, 43

pt_real
 Classes for describing a parser argument, 43

pt_string
 Classes for describing a parser argument, 43

publish
 OPC related functions and classes, 41

push
 Device support for TwinCAT/ADS, 19

Read
 plc::DataValue, 179–181

read_only
 plc, 104

read_ptr
 TcComms::tcProcWrite, 319

read_write
 plc, 104

ReadBinary
 plc::DataValue, 182

records
 plc::BasePLC, 136

replacement_table
 Utility functions and classes, 67

req
 TcComms::tcProcWrite, 320

reserve
 plc::BasePLC, 135

reset_and_read
 plc, 104, 105

RouterCall
 TcComms, 112
 TcComms::AmsRouterNotification, 116

ScannerProc
 plc, 105

scannerThread
 plc, 106

set
 ParseTpy::ads_routing_info, 115
 ParseUtil::memory_location, 269

set_addr
 TcComms::TcPLC, 315

set_ads_state
 TcComms::TcPLC, 315

set_section
 ParseUtil::memory_location, 269

setup
 InfoPlc::BaseInfoltem, 127
 InfoPlc::HistoryInfoltem, 256
 InfoPlc::SimpleInfoltem, 291

SetValid
 plc::DataValue, 183

silent
 OPC related functions and classes, 41

simple
 Classes for describing a type, 33

split_io_support
 EpicsTpy::split_io_support, 294, 295

split_string
 utility functions and classes, 60

start
 InfoPlc::TimeInfoltem, 323
 plc::BasePLC, 135

stat_list
 InfoPlc, 95

std::atomic< string >, 117

std::atomic< wstring >, 118

std::atomic_string< stringT >, 120
 operator=, 121

std::case_char_traits, 158
 compare, 159
 eq, 159
 lt, 160
 ne, 160

std::case_wchar_traits, 161
 compare, 162
 eq, 162
 lt, 163
 ne, 163

std::std::hash< std::stringcase >, 251
 operator(), 251

std::std::hash< std::wstringcase >, 252
 operator(), 252

stdafx.cpp, 373

stdafx.h, 373

stringcase

utility functions and classes, 59
stringcase.h, 374
stringcase_hash.h, 375
stringinval
 Device support for TwinCAT/ADS, 17
stringoutval
 Device support for TwinCAT/ADS, 17
strncasecmp
 utility functions and classes, 60
structtype
 Classes for describing a type, 33
symbol_list
 Classes for describing a symbol, 34
syminfo_processing, 300

tag_processing
 ParseUtil::tag_processing, 304, 305
tc_epics_conv
 Utility functions and classes, 68
tcAlias
 Functions called by the EPICS base, 25
TCatInterface
 TcComms::TCatInterface, 309
TcComms, 109
 ADScallback, 110
 compByOffset, 111
 errorPrintf, 111
 RouterCall, 112
tcComms.cpp, 376
tcComms.h, 377
TcComms::AmsRouterNotification, 116
 RouterCall, 116
TcComms::DataPar, 165
TcComms::TCatInterface, 307
 printTCatVal, 310
 TCatInterface, 309
TcComms::TcPLC, 311
 ADScallback, 317
 closePort, 314
 cyclesLeft, 317
 optimizeRequests, 315
 set_addr, 315
 set_ads_state, 315
 update_scanner, 316
TcComms::tcProcWrite, 317
 add, 319
 read_ptr, 319
 req, 320
tcList
 Functions called by the EPICS base, 25
tcLoadRecords
 Functions called by the EPICS base, 26
tcMacro
 Functions called by the EPICS base, 26
tcPrintVals
 Functions called by the EPICS base, 26
tcSetScanRate
 Functions called by the EPICS base, 27
the_register_devsup

Device support for TwinCAT/ADS, 20
to_epics
 EpicsTpy::epics_conversion, 212
to_filename
 EpicsTpy::epics_macrofiles_processing, 241
TpToEpics.cpp, 379
TpToEpics.h, 379
TpToEpicsConst.h, 381
trim_space
 utility functions and classes, 61, 62
type_enum
 Classes for describing a type, 32
type_multipmap
 Classes for describing a type, 32

unknown
 Classes for describing a type, 33
update_scanner
 TcComms::TcPLC, 316

upper_case
 Utility functions and classes, 68
user_data_set_valid
 plc::BasePLC, 135
UserGetValid
 plc::BaseRecord, 149
 plc::DataValue, 184
UserPush
 plc::BaseRecord, 149
UserRead
 plc::BaseRecord, 150–152
 plc::DataValue, 184–186
UserReadBinary
 plc::BaseRecord, 152
 plc::DataValue, 187
UserSetValid
 plc::BaseRecord, 153
 plc::DataValue, 188
UserWrite
 plc::BaseRecord, 154, 155
 plc::DataValue, 188–190
UserWriteBinary
 plc::BaseRecord, 156
 plc::DataValue, 191
Utility functions and classes, 64, 67
 case_type, 67
 closed, 68
 io_filestat, 68
 ligo_std, 68
 ligo_vac, 68
 lower_case, 68
 no_conversion, 68
 no_dot, 68
 preserve_case, 68
 replacement_table, 67
 tc_epics_conv, 68
 upper_case, 68
utility functions and classes, 59
 split_string, 60
 stringcase, 59

strncasecmp, 60
trim_space, 61, 62
wcsncasewcmp, 62
wstringcase, 59

value_ait_type
 Device support for TwinCAT/ADS, 20

value_count
 Device support for TwinCAT/ADS, 21

waveformval
 Device support for TwinCAT/ADS, 17

wcsncasewcmp
 utility functions and classes, 62

Write
 plc::DataValue, 192–195

write_and_test
 plc, 107, 108

write_only
 plc, 104

WriteBinary
 plc::DataValue, 195

wstringcase
 utility functions and classes, 59

XML tpy file constants, 36